The Virgin Computer Games Series

Series Editor: Tim Hartnell

GAMES FOR YOUR TI 99/4A

££££'s of entertaining games for only £2.95

By Andrew Nelson

GAMES FOR YOUR TI 99/4A

By Andrew Nelson

GAMES FOR YOUR TI 99/4A

By
Andrew Nelson

First published in Great Britain in 1984 by Virgin Books Ltd, 61-63 Portobello Road, London W11 3DD.

ISBN 0 86369 030 0

Printed and bound in Great Britain by Richard Clay (The Chaucer Press) Ltd, Suffolk.

Production services by Book Production Consultants, Cambridge.

Designed by Ray Hyden.

Illustrated by Sue Walliker.

Typeset by QV Typesetting.

Distributed by Arrow Books.

TIM HARTNELL — THE SERIES EDITOR

Tim Hartnell is a leading computer expert and journalist, who has written more than 40 books on computers. His most popular books include *Programming the ZX Spectrum*, *The Giant Book of Computer Games* and *How to Program the IBM PC*.

ANDREW NELSON — THE AUTHOR

Andrew Nelson is interested in many aspects of game-playing, both with computers and with other people. He combined his interests in his first book, *Creating ADVENTURE Programs on your Computer* (Interface Publications, 1983). Andrew is the author of *More Games for Your VIC 20* for this series.

SUE WALLIKER — THE ILLUSTRATOR

Sue Walliker is a freelance illustrator.

ACKNOWLEDGEMENTS

The author would like to acknowledge the assistance of Robert Polack, manager of the microcomputer shop *Gametronics*, who provided invaluable assistance and advice. It was through Robert that the author met the major contributors to this book, Danny Olesh and Damon Pillinger. Others who contributed include James Turner, Wayne Southwick, Tim Hartnell and Kevin Burfitt.

CONTENTS

Editor's Introduction

Typing in a computer program is like opening an unknown door. You do not know until you actually open the door — or, in our case, run the program — what experience is waiting for you. Of course, the sign on the door has given you some indication, but nothing can equal first-hand experience.

You do not know precisely what experiences are waiting for you in the great programs in this book. Of course, if the introduction says you're entering a space game, it's very likely the program won't play 'Guess My Number' when you get it up and running. But the listing rarely hints at the computer's game-playing strategy, or the screen display, or the fun that is waiting for you.

This book has a number of unknown doors — doors leading into outer space and into the fiendish worlds of computer intelligence, wizards and Adventure.

We've provided the doors...and the keys. All you have to do to turn the lock is type in the program, and run it. Whatever you find behind each door, I guarantee you won't be disappointed.

Tim Hartnell
Series editor
London
March 1984

Author's Introduction

The TI 99/4A computer is a machine which has, in my opinion, been underrated for far too long. Featuring superb sound, user-definable graphics and a good keyboard, the TI 99/4A is a great computer to use.

Although some of its commands are a little non-standard, they are easily mastered and can be used most effectively, as I hope I have demonstrated in this book.

To ensure that this book is of maximum use to you, the standard onboard BASIC has been adhered to in all except one of the programs. If you think that the standard language is too limited for worthwhile programming, you've got a pleasant surprise in store.

I'm sure you will enjoy running the programs in this book, and I hope you have fun adapting and improving them.

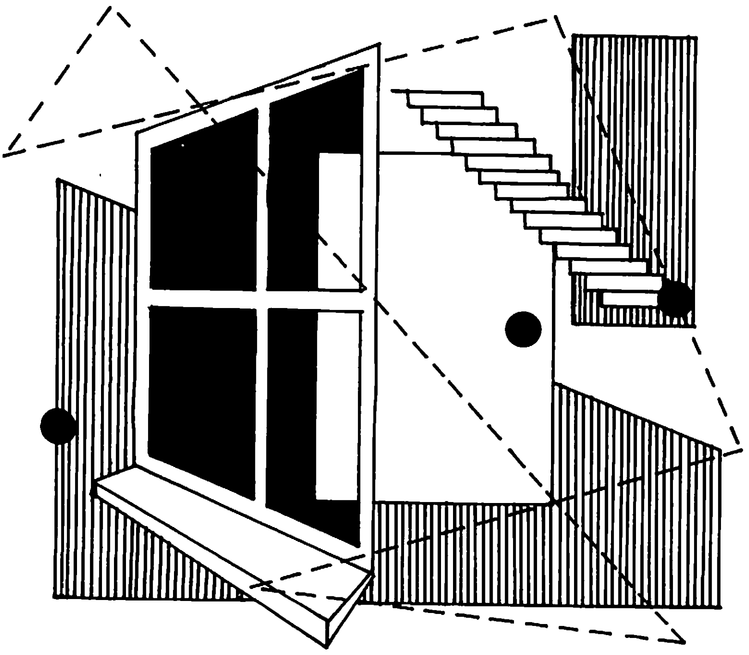
Andrew Nelson
London
January 1984

3-D MAZE

In this game, you have to try and find a secret room hidden within a vast mansion, designed by the mad architect, T.I. McNinetyniner. Although the maze remains the same from game to game, the location of the room you are searching for changes each time you run the program.

The room is presented graphically in front of you, creating quite an uncanny feeling of 'reality' as the walls move past you. You move using the arrow keys.

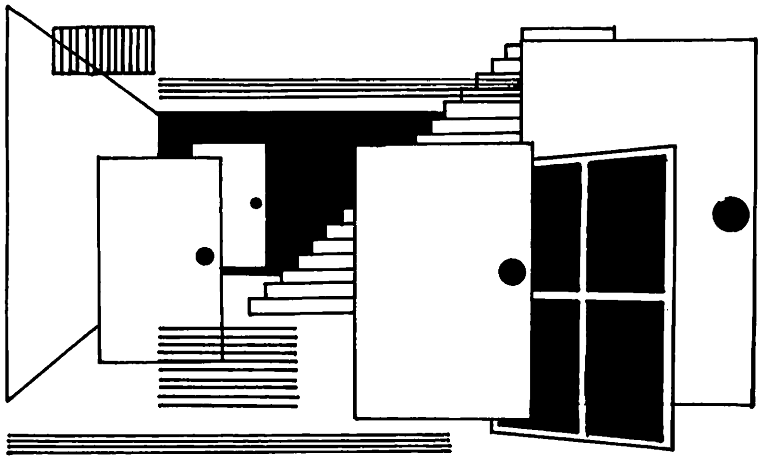
You will be asked to enter your choice of colours when you run the program. Start by entering 14,2 which works very well. Then you can choose your own colour combinations on subsequent runs.



```
10 REM  3-D MAZE
20 REM
30 INPUT "FOREGROUND,BACKGROUND COLOR ?":FG,BG
40 DIM ROOM(10,10),HUH(11,4),C(11,4)
50 GOSUB 680
60 A$(1)="SOUTH"
70 A$(2)="WEST"
80 A$(3)="NORTH"
90 A$(4)="EAST"
100 FOR X=1 TO 10
110 FOR Y=1 TO 10
120 READ ROOM(X,Y)
130 NEXT Y
140 NEXT X
150 FOR I=1 TO 11
160 READ HUH(I,3),HUH(I,4),HUH(I,1),HUH(I,2)
170 NEXT I
180 XX=1
190 YY=1
200 FACE=1
210 FOR I=1 TO 11
220 READ C(I,1),C(I,2),C(I,3),C(I,4)
230 NEXT I
240 REM
250 REM   ^SETUP,RUNv
260 GOTO 390
270 CALL SOUND(-500,110,0)
280 CALL KEY(1,K,S)
290 IF S=0 THEN 280
300 K=-(K=5)-2*(K=3)-3*(K+1=1)-4*(K=2)
310 K=K-(K=0)
320 FACE=FACE+(K=4)-2*(K=3)-(K=2)
330 FACE=FACE+4*(FACE>4)-4*(FACE<1)
340 IF K=1 THEN 360
350 GOTO 390
360 IF HUH(ROOM(XX,YY),FACE)=0 THEN 270
370 XX=XX+(FACE=3)-(FACE=1)
380 YY=YY+(FACE=2)-(FACE=4)
390 R=C(ROOM(XX,YY),FACE)
400 CALL CLEAR
410 REM  PRINT UP INFO
420 A=1
430 IF (R=1)+(R=2)+(R=4)+(R=7)THEN 450
440 A=2
450 GOSUB 890
460 A=7
470 IF (R=2)+(R=3)+(R=6)+(R=9)THEN 490
480 A=8
490 GOSUB 890
```

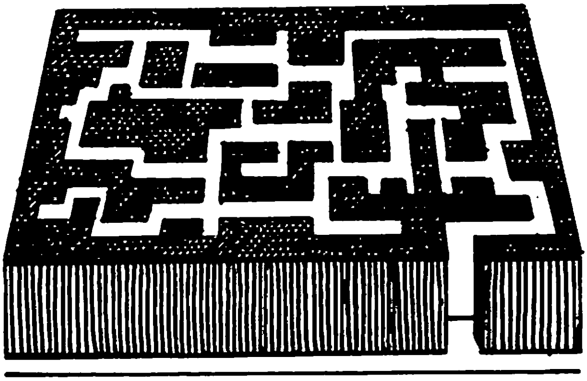
```
500 A=11
510 IF (R=4)+(R=5)+(R=6)+(R=10)THEN 520
                                   ELSE 540
520 GOSUB 890
530 GOTO 280
540 R=C(ROOM(XX+(FACE=3)-(FACE=1),
     YY+(FACE=2)-(FACE=4)),FACE)
550 A=3
560 IF (R=1)+(R=2)+(R=4)+(R=7)THEN 580
570 A=4
580 GOSUB 890
590 A=5
600 IF (R=2)+(R=3)+(R=6)+(R=9)THEN 620
610 A=6
620 GOSUB 890
630 A=10
640 IF (R=4)+(R=5)+(R=6)+(R=10)THEN 660
650 A=9
660 GOSUB 890
670 GOTO 280
680 REM  SEND THE 'SETUP' TO HERE
690 A$(1)="80C0E0F0F8FCFEFF"
700 A$(2)="FFFEFCF8F0E0C080"
710 A$(3)="FF7F3F1F0F070301"
720 A$(4)="0103070F1F3F7FFF"
730 A$(5)="FFFFFFFFFFFFFFFF"
740 A$(6)="0"
750 CALL CHAR(139,A$(1))
760 CALL CHAR(140,A$(2))
770 CALL CHAR(141,A$(3))
780 CALL CHAR(142,A$(4))
790 CALL CHAR(143,A$(5))
800 CALL CHAR(144,A$(6))
810 CALL COLOR(14,FG,1)
820 CALL SCREEN(BG)
830 IF BG>2 THEN 870
840 FOR I=1 TO 13
850 CALL COLOR(I,16,BG)
860 NEXT I
870 REM   139=\B,140=\T,141=/T,142=/B,
                   143=FULL,144=BLANK
880 RETURN
890 RE I  SEND THE MAP TO HERE
900 IF (A<1)+(A>12)THEN 920
910 ON A GOSUB 930,1000,1070,1140,1210,
     1280,1350,1420,1490,1610,1660,1710
920 RETURN
930 REM  LHS FRONT SOLID, COLOR=N
```

```
940 FOR I=1 TO 6
950 CALL HCHAR(I,I,139)
960 CALL HCHAR(25-I,I,140)
970 CALL VCHAR(I+1,I,143,24-2*I)
980 NEXT I
990 RETURN
1000 REM  LHS FRONT DOOR, COLOR=N
1010 FOR I=1 TO 6
1020 CALL VCHAR(6,I,143,14)
1030 NEXT I
1040 CALL HCHAR(6,6,139)
1050 CALL HCHAR(19,6,140)
1060 RETURN
1070 REM  LHS SIDE WALL,COLOR=N
1080 FOR I=7 TO 10
1090 CALL HCHAR(I,I,139)
1100 CALL HCHAR(25-I,I,140)
1110 CALL VCHAR(I+1,I,143,24-I*2)
1120 NEXT I
1130 RETURN
1140 REM  LHS SIDE DOOR, COLOR=N
1150 FOR I=7 TO 10
1160 CALL VCHAR(10,I,143,6)
1170 NEXT I
1180 CALL HCHAR(10,10,139)
1190 CALL HCHAR(15,10,140)
1200 RETURN
1210 REM  RHS SIDE WALL,COLOR=N
1220 FOR I=15 TO 18
1230 CALL HCHAR(I,I,141)
1240 CALL HCHAR(25-I,I,142)
1250 CALL VCHAR(26-I,I,143,I*2-26)
1260 NEXT I
1270 RETURN
1280 REM  RHS SIDE DOOR,COLOR=N
1290 FOR I=15 TO 18
1300 CALL VCHAR(10,I,143,6)
1310 NEXT I
1320 CALL HCHAR(10,15,142)
1330 CALL HCHAR(15,15,141)
1340 RETURN
1350 REM  RHS FRONT WALL,COLOR=N
1360 FOR I=19 TO 24
1370 CALL HCHAR(I,I,141)
1380 CALL HCHAR(25-I,I,142)
1390 CALL VCHAR(26-I,I,143,I*2-26)
1400 NEXT I
1410 RETURN
1420 REM  RHS FRONT DOOR,COLOR=N
```



```
1430 FOR I=19 TO 24
1440 CALL VCHAR(6,I,143,14)
1450 NEXT I
1460 CALL HCHAR(6,19,142)
1470 CALL HCHAR(19,19,141)
1480 RETURN
1490 REM  CENTER FOREVER
1500 FOR I=11 TO 12
1510 CALL HCHAR(I,I,139)
1520 CALL HCHAR(25-I,I,140)
1530 NEXT I
1540 FOR I=13 TO 14
1550 CALL HCHAR(I,I,141)
1560 CALL HCHAR(25-I,I,142)
1570 NEXT I
1580 CALL VCHAR(12,11,143,2)
1590 CALL VCHAR(12,14,143,2)
1600 RETURN
1610 REM  CENTER WALL
1620 FOR I=10 TO 15
1630 CALL VCHAR(10,I,143,6)
1640 NEXT I
1650 RETURN
1660 REM  MIDDLE 'BIG' WALL
1670 FOR I=6 TO 19
1680 CALL VCHAR(6,I,143,14)
```

```
1690 NEXT I
1700 RETURN
1710 CALL CLEAR
1720 RETURN
1730 DATA 4,5,5,5,5,5,5,5,5,6
1740 DATA 7,8,9,7,8,9,7,8,9,2
1750 DATA 7,1,7,9,1,7,9,4,9,2
1760 DATA 1,8,8,8,8,11,8,8,8,8
1770 DATA 1,3,1,3,1,3,1,3,1,3
1780 DATA 4,6,4,6,4,6,4,6,4,6
1790 DATA 1,3,1,3,1,3,1,3,1,3
1800 DATA 1,1,1,1,1,1,1,1,1,1
1810 DATA 1,1,1,1,1,1,1,1,1,1
1820 DATA 1,1,1,1,1,1,1,1,1,1
1830 DATA 1,1,0,0,1,0,1,0,1,0,0,1,0,1,1,0,
          0,1,0,1,0,0,1,1,1,1,1,0,1,1,0,1,
          1,0,1, 1
1840 DATA 0,1,1,1,1,1,1,1
1850 DATA 6,4,1,3,2,5,2,5,4,1,3,6,3,6,4,1,
          5,2,5,2,1,3,6,4,9,10,7,8,10
1860 DATA 7,8,9,7,8,9,10,8,9,10,7,
          11,11,11,11
```



SEARCH FOR THE HOLY GRAIL

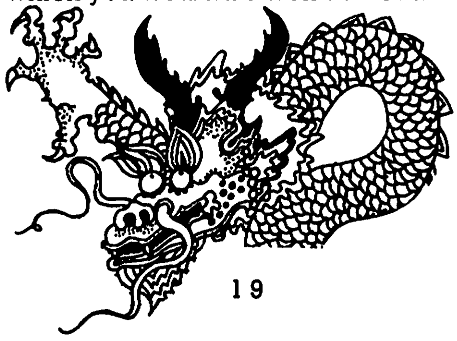
Now you can take on the Monty Python team at their own quest, as you traverse the lonely fields of the TV screen in search of the Grail.

In actual fact, you don't have to search very hard (as the Grail is shown plainly on the screen from the beginning of the game). However, knowing where it is and actually getting to it are two quite different things. You have to get to the Grail through a maze of walls built of odd-shaped stones.

You move through the maze using the arrow keys. Your path to the Grail is impeded by a ferocious dragon, hell-bent on your destruction. Thanks to clever programming, the dragon is relatively intelligent. It is after you at all times, and is able to move through walls, so you really haven't got much hope!

If you manage to get to the Grail on the first screen shown, a new maze will be created. However, this second maze is invisible. All you will see on the screen is the Grail, yourself, and the dragon.

As an extra twist, a bouncing ball will appear on the screen which you would be well advised to beware.

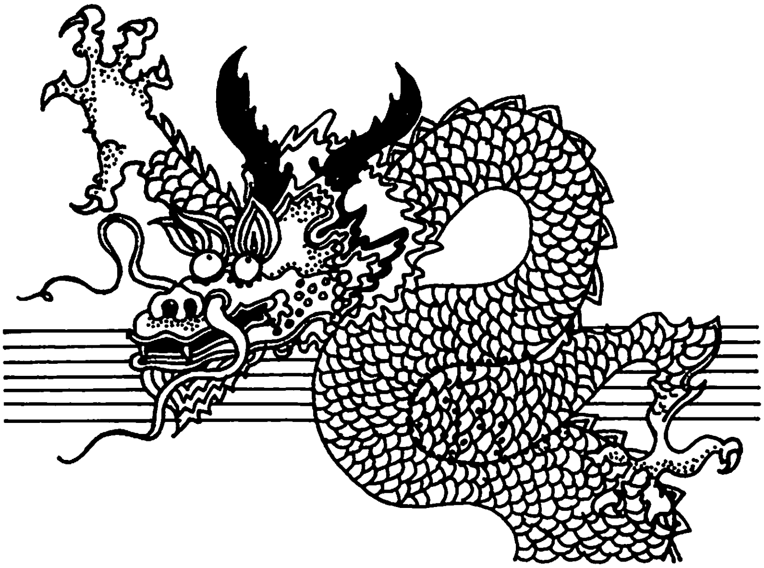


```
10 REM  SEARCH FOR THE HOLY GRAIL
20 GAMES=1
30 TOTS=387
40 HS=387
50 RANDOMIZE
60 CALL SCREEN(2)
70 SPEED=4
80 Y=23
90 A=8
100 B=8
110 R=10
120 C=16
130 X=2
140 CALL CHAR(131,"FFC3A59999A5C3FF")
150 OP=0
160 X$(0)="0000001"
170 X$(1)="0000001818"
180 X$(2)="0000183C3C18"
190 X$(4)="3C7EFFFFFFFF7E3C"
200 X$(3)="00183C7E7E3C18"
210 X$(5)=X$(4)
220 X$(6)=X$(4)
230 X$(7)=X$(3)
240 CALL CHAR(145,X$(0))
250 X$(8)=X$(2)
260 X$(9)=X$(1)
270 CALL CLEAR
280 CALL HCHAR(24,1,131,64)
290 CALL VCHAR(1,32,131,48)
300 CALL HCHAR(11,16,138)
310 CALL HCHAR(3,3,131,13)
320 CALL HCHAR(3,17,131,10)
330 CALL VCHAR(3,15,131,9)
340 CALL CHAR(133,"00101038101")
350 CALL VCHAR(3,17,131,9)
360 CALL HCHAR(12,15,131,3)
370 CALL HCHAR(2,2,133,1)
380 CALL HCHAR(9,5,131,10)
390 CALL HCHAR(11,5,131,10)
400 CALL HCHAR(10,15,32,4)
410 CALL VCHAR(5,3,131,10)
420 CALL VCHAR(3,30,131,10)
430 CALL HCHAR(22,2,131,8)
440 CALL CHAR(138,"00FF7E3C18187E7E")
450 CALL HCHAR(22,11,131,20)
460 CALL HCHAR(20,5,131,6)
470 CALL HCHAR(20,15,131,10)
480 CALL HCHAR(18,6,131,25)
490 CALL HCHAR(16,3,131,4)
```

```
500 CALL HCHAR(16,10,131,10)
510 CALL VCHAR(19,12,131,3)
520 CALL HCHAR(6,4,131,8)
530 CALL HCHAR(11,14,32)
540 CALL CHAR(144,"60E2151838181838")
550 CALL HCHAR(7,3,32)
560 CALL HCHAR(6,2,131,6)
570 CALL VCHAR(5,19,131,12)
580 CALL VCHAR(18,13,32)
590 CALL VCHAR(12,8,131,6)
600 CALL VCHAR(17,3,131,4)
610 CALL VCHAR(20,5,131,3)
620 CALL HCHAR(11,22,133)
630 CALL HCHAR(14,20,131,5)
640 CALL CHAR(150,"003C3C183C3C2424")
650 CALL HCHAR(12,22,131,6)
660 CALL HCHAR(15,8,32)
670 CALL HCHAR(14,10,131,3)
680 CALL VCHAR(14,16,131,2)
690 CALL HCHAR(20,12,32)
700 CALL HCHAR(15,19,32)
710 CALL VCHAR(12,26,131,6)
720 CALL HCHAR(16,4,129)
730 CALL HCHAR(8,15,129)
740 CALL HCHAR(16,15,129)
750 CALL HCHAR(3,14,129)
760 CALL HCHAR(11,7,129,3)
770 CALL HCHAR(19,26,133)
780 FOR I=1 TO 12
790 CALL COLOR(I,16,1)
800 NEXT I
810 CALL COLOR(13,6,1)
820 CALL COLOR(14,11,1)
830 CALL COLOR(15,10,1)
840 CALL COLOR(16,4,1)
850 CALL CHAR(129,"FFC3A59999A5C3FF")
860 CALL HCHAR(Y,X,150)
870 FOR I=1 TO 19-SPEED+(SCORE/15)
880 S=INT(RND*18)+3
890 T=INT(RND*26)+3
900 IF (S=11)*(T=16)THEN 880
910 CALL HCHAR(S,T,129-2*(RND>.8))
920 NEXT I
930 CALL HCHAR(INT(RND*22)+2,
                     INT(RND*30)+2,133)
940 RR=R+(R>Y)-(R<Y)
950 CC=C+(C>X)-(C<X)
960 CALL GCHAR(R,CC,K)
```

```
970 IF (K=131)+(K=145)+
         (K=138)+(K=144)THEN 1020
980 IF K=150 THEN 1590
990 CALL HCHAR(R,C,32)
1000 CALL HCHAR(R,CC,144)
1010 C=CC
1020 CALL GCHAR(RR,C,K)
1030 IF (K=131)+(K=145)+(K=138)+
                     (K=144)THEN 1080
1040 IF K=150 THEN 1590
1050 CALL HCHAR(R,C,32)
1060 CALL HCHAR(RR,C,144)
1070 R=RR
1080 FOR I=1 TO SPEED
1090 CALL KEY(1,K,S)
1100 IF (S=0)+(K>5)+(K=1)THEN 1300
1110 XX=X+(K=2)-(K=3)
1120 YY=Y+(K=5)-(K+1=1)
1130 CALL GCHAR(YY,XX,K)
1140 IF K<>133 THEN 1190
1150 CALL SOUND(20,440,0)
1160 CALL SOUND(40,480,0)
1170 CALL SOUND(20,440,0)
1180 SCORE=SCORE+5
1190 SCORE=SCORE-(K=131)*(SCORE>0)
1200 IF (K<>129)*(K<>131)THEN 1220
1210 CALL SOUND(-10,440,0,-7,7)
1220 IF (K=129)+(K=131)+(K=150)THEN 1300
1230 IF (K=145)+(K=144)THEN 1590
1240 I=I+9*(K=133)
1250 IF K=138 THEN 1420
1260 CALL HCHAR(Y,X,32)
1270 CALL HCHAR(YY,XX,150)
1280 Y=YY
1290 X=XX
1300 NEXT I
1310 IF OP THEN 1390
1320 CALL HCHAR(A,B,32)
1330 A=INT(RND*22)+2
1340 B=INT(RND*30)+2
1350 IF (A=11)*(B=16)THEN 1330
1360 OP=10
1370 CALL HCHAR(A,B,145)
1380 GOTO 940
1390 OP=OP-1
1400 CALL CHAR(145,X$(OP))
1410 GOTO 940
1420 SPEED=SPEED+(SPEED>1)
```

```
1430 FOR I=1 TO 15
1440 CALL SOUND(-100,110+20*I,0)
1450 NEXT I
1460 SCORE=SCORE+20
1470 IF SPE=1 THEN 1540
1480 IF SPEED<>3 THEN 1580
1490 PRINT :::::::::::::::::::::::::::::
 ::::::::::::::::::"      BONUS LEVEL":::
:::::::::::::::::::::::::::::::::
1500 CALL SCREEN(6)
1510 IF SPE=1 THEN 1540
1520 SPE=1
1530 GOTO 80
```



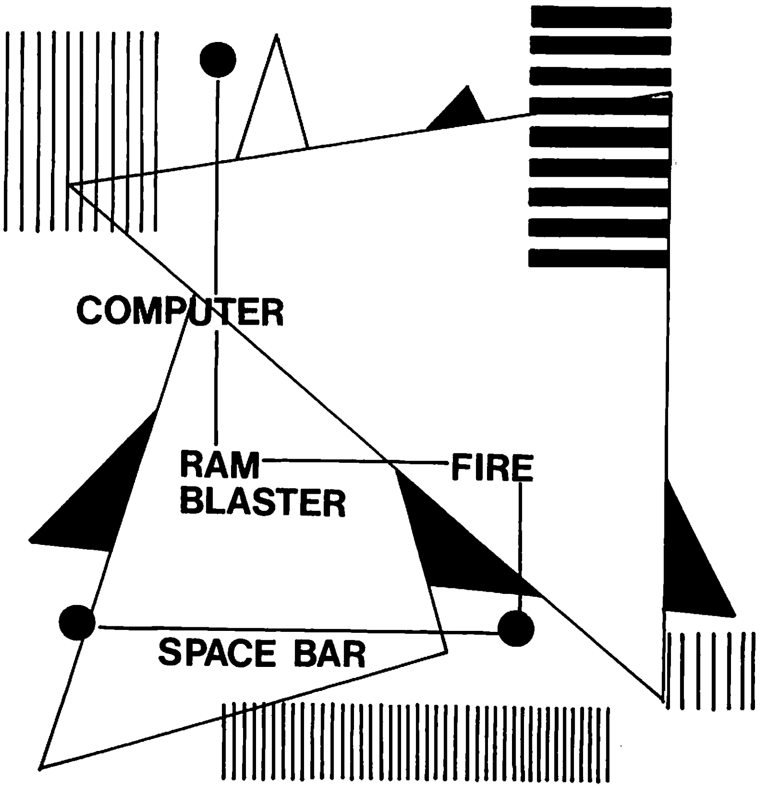
```
1540 CALL SCREEN(2)
1550 SPEED=2
1560 SCORE=SCORE+15
1570 SPE=0
1580 GOTO 80
1590 CALL SOUND(100,440,0)
1600 CALL SOUND(20,440,30)
1610 CALL SOUND(100,340,0)
1620 CALL SOUND(20,340,30)
```

```
1630 CALL SOUND(100,340,0)
1640 CALL SOUND(20,380,30)
1650 CALL SOUND(100,380,0)
1660 CALL SOUND(20,330,30)
1670 CALL SOUND(100,330,0)
1680 CALL SOUND(80,380,30)
1690 CALL SOUND(100,380,0)
1700 CALL SOUND(20,430,30)
1710 CALL SOUND(100,410,0)
1720 GAMES=GAMES+1
1730 PRINT ::"SCORE --->";SCORE
1740 TOTS=TOTS+SCORE
1750 PRINT "HIGH SCORE ===>";HS
1760 PRINT ::"AVERAGE SCORE ==>":
   "FROM ";GAMES;" GAMES":"=";TOTS/GAMES:
1770 IF SCORE>HS THEN 1840
1780 INPUT "PLAY AGAIN ?":A$
1790 IF SEG$(A$,1,1)<>"Y" THEN 1830
1800 SCORE=0
1810 SPE=0
1820 GOTO 50
1830 STOP
1840 HS=SCORE
1850 GOTO 1780
```

RAM BLASTER

You are the dreaded RAM-rammer, trying to destroy all the computer components on the screen.

When you first run the game, you'll see the points you can score for hitting each kind of component. You move using the arrow keys, and fire by pressing the space bar. Note that you can only fire by pressing the space bar *as* you move. The little black bullet always moves in the direction you are currently travelling.



```
10 REM   RAM BLASTER
20 REM  BY D&D
30 RANDOMIZE
40 LEVEL=1
50 CALL CLEAR
60 PRINT "         ALL SYSTEMS GO":
   :"          STANDBY":::::::
70 FOR I=127 TO 159
80 CALL CHAR(I,STR$(INT(RND*999999999)+1))
90 NEXT I
100 KILL=0
110 A=9
120 X=10
130 DIM SC(5)
140 SC(1)=40
150 SC(2)=70
160 SC(3)=100
170 SC(4)=200
180 XM=0
190 CALL CLEAR
200 DIM HS(20),HS$(20)
210 GOSUB 2840
220 Y=10
230 CALL CHAR(105,"FF81C399C381FF")
240 DIM XX(10),YY(10)
250 CALL CHAR(104,"0000003C3C000000")
260 SC(5)=220
270 CALL CHAR(127,"")
280 YM=1
290 LEVEL=1
300 SCORE=0
310 CALL CHAR(103,"00183C7E7E3C18")
320 DIM T(10)
330 FOR I=1 TO A
340 T(I)=INT(RND*3)+1
350 XX(I)=INT(RND*24)+1
360 YY(I)=INT(RND*32)+1
370 NEXT I
380 CALL CHAR(100,"7E3C7E3C7E3C7E")
390 CALL CHAR(101,"0000FFE7E73C3C")
400 CALL CHAR(102,"0000FF421824FF")
410 FREE=1
420 CALL KEY(3,K,S)
430 LIFE=LIFE+1
440 IF LIFE>3 THEN 1000
450 QQ=12
460 FOR I=1 TO 16
470 CALL COLOR(I,02,QQ)
480 NEXT I
```

```
490 CALL COLOR(9,05,QQ)
500 CALL COLOR(10,2,QQ)
510 AX=22
520 XX$="    "&STR$(4-LIFE)&" LIVES     "
530 GOSUB 3220
540 KILL=0
550 AX=23
560 XX$="LEVEL -->"&STR$(LEVEL)
570 GOSUB 3220
580 AX=24
590 XX$="PRESS ANY KEY TO BEGIN LEVEL"
600 GOSUB 3220
610 CALL KEY(0,K,S)
620 IF S=0 THEN 610
630 AX=10
640 XX$="GET READY"
650 GOSUB 3220
660 AX=11
670 XX$=" GET SET "
680 GOSUB 3220
690 AX=13
700 XX$="              GO           "
710 GOSUB 3220
720 CALL SCREEN(QQ)
730 CALL CLEAR
740 CALL KEY(0,KEY,STATUS)
750 IF KEY=32 THEN 1970
760 XM=(XM-(KEY=ASC("X"))+(KEY=ASC("E")))*
     (KEY<>ASC("S"))*(KEY<>ASC("D"))
770 YM=(YM-(KEY=ASC("D"))+(KEY=ASC("S")))
     *(KEY<>ASC("E"))*(KEY<>ASC("X"))
780 CALL HCHAR(X,Y,32)
790 X=X+XM
800 Y=Y+YM
810 X=X-24*(X<=0)
820 X=X+24*(X>=25)
830 Y=Y-32*(Y<=0)
840 Y=Y+32*(Y>=33)
850 CALL GCHAR(X,Y,HIT)
860 IF HIT=32 THEN 890
870 GOSUB 1810
880 GOTO 990
890 CALL HCHAR(X,Y,100)
900 ABC=N
910 GOTO 930
920 IF N=ABC THEN 2330
930 N=N+1
940 N=N+A*(N>A)
950 IF T(N)=0 THEN 920
```

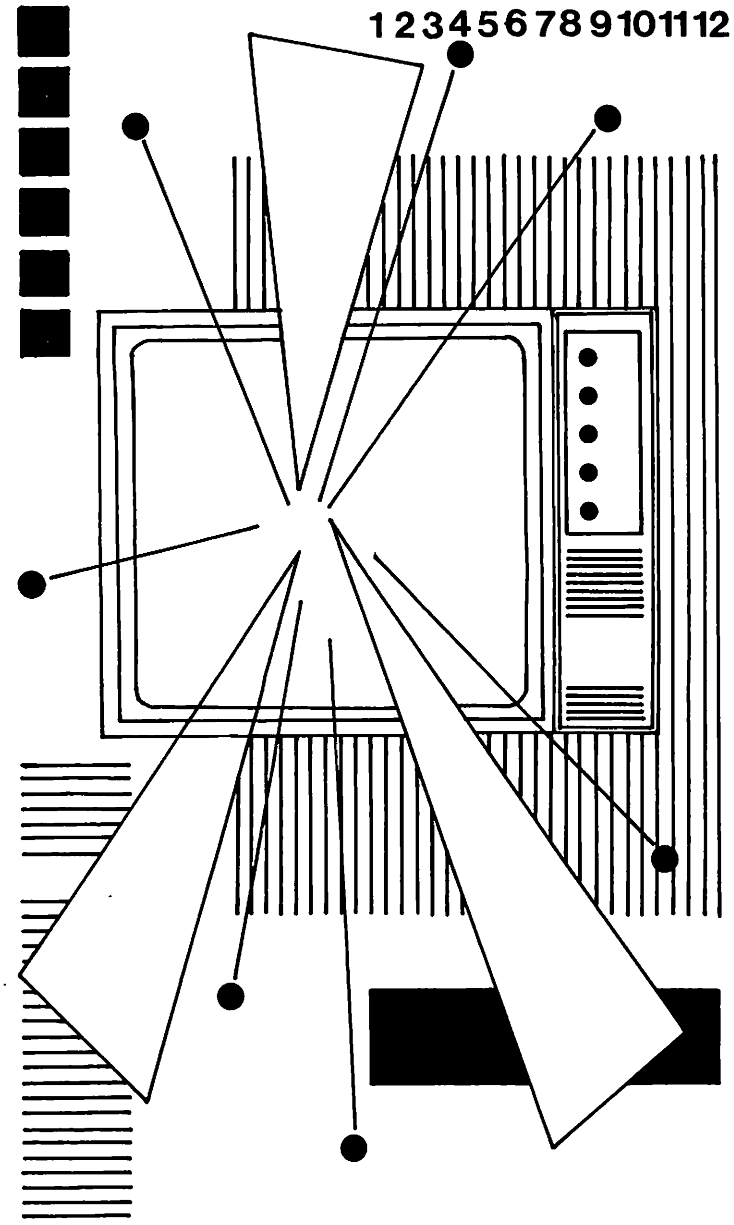
```
960 XXX=1
970 ON T(N)GOSUB 1030,1180,1290,1310,1420
980 IF XXX=1 THEN 1010
990 GOTO 430
1000 GOTO 1950
1010 CALL SOUND(1,1440,0)
1020 GOTO 740
1030 S=(XX(N)>X)-(XX(N)<X)
1040 REM  S=S+(S=0)
1050 D=(S=0)*((YY(N)<Y)-(YY(N)>Y))
1060 D=D+10000
1070 CALL HCHAR(XX(N),YY(N),32)
1080 XX(N)=XX(N)+S
1090 YY(N)=YY(N)+D
1100 XX(N)=XX(N)+(XX(N)>24)-(XX(N)<=0)
1110 YY(N)=YY(N)+(YY(N)>32)-(YY(N)<=0)
1120 CALL GCHAR(XX(N),YY(N),HIT)
1130 IF HIT=32 THEN 1160
1140 GOSUB 1820
1150 RETURN
1160 CALL HCHAR(XX(N),YY(N),101)
1170 RETURN
1180 CALL HCHAR(XX(N),YY(N),32)
1190 XX(N)=XX(N)+(XX(N)>X)-(XX(N)<X)
1200 YY(N)=YY(N)+(YY(N)>Y)-(YY(N)<Y)
1210 XX(N)=XX(N)+(XX(N)>24)-(XX(N)<=0)
1220 YY(N)=YY(N)+(YY(N)>32)-(YY(N)<1)
1230 CALL GCHAR(XX(N),YY(N),HIT)
1240 IF HIT=32 THEN 1270
1250 GOSUB 1820
1260 RETURN
1270 CALL HCHAR(XX(N),YY(N),102)
1280 RETURN
1290 IF (SQR((XX(N)-X)^2)>=3)+
           (SQR((YY(N)-Y)^2)>=3)THEN 1030
1300 T(N)=4
1310 CALL HCHAR(XX(N),YY(N),32)
1320 XX(N)=XX(N)+(XX(N)>(X+XM))-
                        (XX(N)<(X+XM))
1330 YY(N)=YY(N)+(YY(N)>(Y+YM)
                        -(YY(N)<(Y+YM))
1340 XX(N)=XX(N)+(XX(N)>24)-(XX(N)<1)
1350 YY(N)=YY(N)+(YY(N)>32)-(YY(N)<1)
1360 CALL GCHAR(XX(N),YY(N),HIT)
1370 IF HIT=32 THEN 1400
1380 GOSUB 1820
1390 RETURN
1400 CALL HCHAR(XX(N),YY(N),103)
1410 RETURN
```

```
1420 FOR LOOP=1 TO 2
1430 IF INT(RND*10)+1>1 THEN 1460
1440 CALL HCHAR(XX(N),YY(N),104)
1450 GOTO 1470
1460 CALL HCHAR(XX(N),YY(N),32)
1470 XX(N)=XX(N)+(XX(N)>X)-(XX(N)<X)
1480 YY(N)=YY(N)+(YY(N)>Y)-(YY(N)<Y)
1490 XX(N)=XX(N)+(XX(N)>24)-(XX(N)<1)
1500 YY(N)=YY(N)+(YY(N)>32)-(YY(N)<1)
1510 CALL GCHAR(XX(N),YY(N),HIT)
1520 IF HIT=32 THEN 1550
1530 GOSUB 1820
1540 RETURN
1550 IF (SQR((XX(N)-X)^2)<5)*(SQR((YY(N)
     -Y)^2)<5)THEN 1580
1560 CALL HCHAR(XX(N),YY(N),127)
1570 GOTO 1590
1580 CALL HCHAR(XX(N),YY(N),105)
1590 NEXT LOOP
1600 RETURN
1610 CALL SOUND(-200,-7,0)
1620 IF HIT=104 THEN 1790
1630 SCORE=SCORE+SC(T(N))
1640 T(N)=0
1650 GOSUB 1700
1660 KILL=KILL+1
1670 LLLL=SCORE
1680 IF KILL>=A THEN 2330
1690 RETURN
1700 IF LLLL>=3000*FREE THEN 1780
1710 IF SCORE<3000*FREE THEN 1780
1720 FOR IS=1 TO 5
1730 CALL SOUND(150,440,0)
1740 CALL SOUND(150,40000,30)
1750 NEXT IS
1760 LIFE=LIFE-1
1770 FREE=FREE+1
1780 RETURN
1790 CALL HCHAR(XX(N),YY(N),32)
1800 GOTO 1630
1810 HIT=100
1820 IF HIT<>100 THEN 1610
1830 FOR I=1 TO 30
1840 CALL SOUND(-100,-7,0)
1850 CALL HCHAR(X,Y,INT(RND*32)+128)
1860 NEXT I
1870 PRINT "SCORE=";SCORE:::::
1880 XXX=0
1890 FOR III=1 TO A
```

```
1900 IF T(N)=0 THEN 1930
1910 XX(N)=INT(RND*24)+1
1920 YY(N)=INT(RND*32)+1
1930 NEXT III
1940 RETURN
1950 GOSUB 2590
1960 GOTO 100
1970 YB=YM
1980 XB=XM
1990 PX=X
2000 PY=Y
2010 KONST=10-LEVEL
2020 IF KONST>3 THEN 2040
2030 KONST=4
2040 FOR I=1 TO KONST
2050 PX=PX+XB
2060 PY=PY+YB
2070 IF (PX<1)+(PY<1)+(PX>24)+
      (PY>32)THEN 2170
2080 CALL GCHAR(PX,PY,SPLAT)
2090 IF SPLAT=100 THEN 2140
2100 IF SPLAT=104 THEN 2140
2110 IF SPLAT<>32 THEN 2180
2120 CALL HCHAR(PX,PY,104)
2130 CALL HCHAR(PX,PY,32)
2140 NEXT I
2150 XB=0
2160 YB=0
2170 GOTO 760
2180 FOR N=1 TO A
2190 IF T(N)=0 THEN 2210
2200 IF (XX(N)=PX)*(YY(N)=PY)THEN 2230
2210 NEXT N
2220 GOTO 760
2230 LLLL=SCORE
2240 SCORE=SCORE+SC(T(N))
2250 GOSUB 1700
2260 LLLL=SCORE
2270 T(N)=0
2280 KILL=KILL+1
2290 IF KILL>=A THEN 2330
2300 CALL SOUND(-140,-5,0)
2310 CALL HCHAR(PX,PY,32)
2320 GOTO 760
2330 FOR I=1 TO 10
2340 CALL COLOR(1,QQ,INT(RND*16)+1)
2350 NEXT I
2360 CALL COLOR(1,16,QQ)
2370 LEVEL=LEVEL+1
```

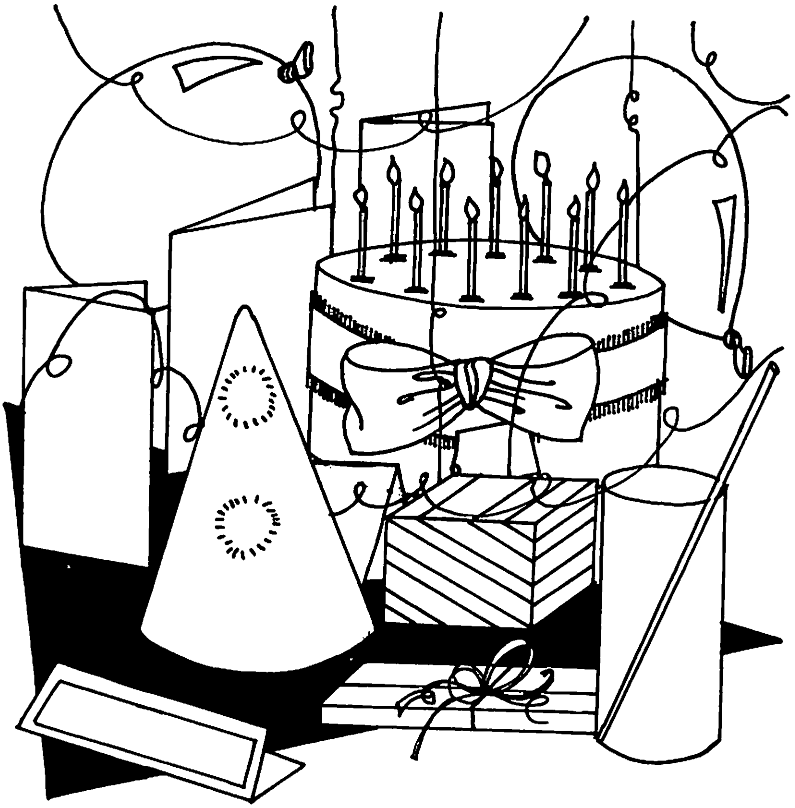
```
2380 FOR I=1 TO A
2390 T(I)=INT((RND*5)+1)
2400 G=G-(T(I)=4)
2410 H=H-(T(I)=5)
2420 XX(I)=INT(RND*24)+1
2430 YY(I)=INT(RND*32)+1
2440 NEXT I
2450 IF (G>LEVEL/3)+(G>A-2)THEN 2490
2460 H=0
2470 G=0
2480 GOTO 2380
2490 X=10
2500 IF ((H+G)>LEVEL/2)+((G+H)>A-2)THEN 2540
2510 G=0
2520 H=0
2530 GOTO 2380
2540 Y=10
2550 XM=0
2560 YM=1
2570 PRINT ::::::::::::"SCORE="
              ,SCORE:::::::::::
2580 GOTO 450
2590 XX$="     GAME OVER        "
2600 FOR I=1 TO LEN(XX$)
2610 CALL HCHAR(12,(15-INT(.5*LEN(XX$)))+I,
                ASC(SEG$(XX$,I,1)))
2620 NEXT I
2630 FOR I=1 TO 20
2640 IF SCORE>HS(I)THEN 2730
2650 NEXT I
2660 FOR I=1 TO 20
2670 PRINT HS(I),HS$(I)
2680 NEXT I
2690 FOR I=1 TO 1000
2700 NEXT I
2710 RETURN
2720 REM
2730 FOR J=20 TO I+1 STEP -1
2740 HS(J)=HS(J-1)
2750 HS$(J)=HS$(J-1)
2760 NEXT J
2770 INPUT "NAME PLEASE ?":HS$(I)
2780 CALL CLEAR
2790 IF LEN(HS$(I))<11 THEN 2820
2800 PRINT "TEN CHARACTERS LONG-NO MORE"
2810 GOTO 2770
2820 HS(I)=SCORE
2830 GOTO 2660
```

```
2840 IF HS(1)>0 THEN 2890
2850 FOR I=1 TO 20
2860 HS(I)=5000-15*I-INT(RND*10)
2870 HS$(I)="RAMCHIP"&STR$(I)
2880 NEXT I
2890 SCORE=0
2900 AX=4
2910 XX$="RAMCHIP"
2920 GOSUB 3220
2930 AX=6
2940 XX$="d - YOUR RAMCHIP"
2950 GOSUB 3220
2960 REM  d,h,e,40,f,70,e,100,g,200,i,220
2970 AX=7
2980 XX$="e - RAMCHIPS     40"
2990 GOSUB 3220
3000 AX=8
3010 XX$="f - ROMCHIPS     70"
3020 GOSUB 3220
3030 AX=9
3040 XX$="e - PROMS,V1    100"
3050 GOSUB 3220
3060 AX=10
3070 XX$="g - PROMS,V2    200"
3080 GOSUB 3220
3090 AX=11
3100 XX$="i - EPROMS      220"
3110 GOSUB 3220
3120 AX=12
3130 XX$="HIDDEN EPROMS  220"
3140 GOSUB 3220
3150 AX=13
3160 XX$="h - STATIC ELECTRICITY -BEWARE"
3170 GOSUB 3220
3180 AX=15
3190 XX$="FREE RAMCHIP EACH 3000 POINTS"
3200 GOSUB 3220
3210 RETURN
3220 AY=(16-(.5*LEN(XX$)))
3230 IF XX$<>"GET READY" THEN 3250
3240 CALL CLEAR
3250 FOR IJ=1 TO LEN(XX$)
3260 CALL HCHAR(AX,AY+IJ,
          ASC(SEG$(XX$,IJ,1)))
3270 NEXT IJ
3280 RETURN
```


1 2 3 4 5 6 7 8 9 10 11 12

HAPPY BIRTHDAY

This simple program, by Damon Pillinger and Mark Charlton, will tell you which day of the week you were born on. You can also use it, of course, to find out on which day historical events occurred.



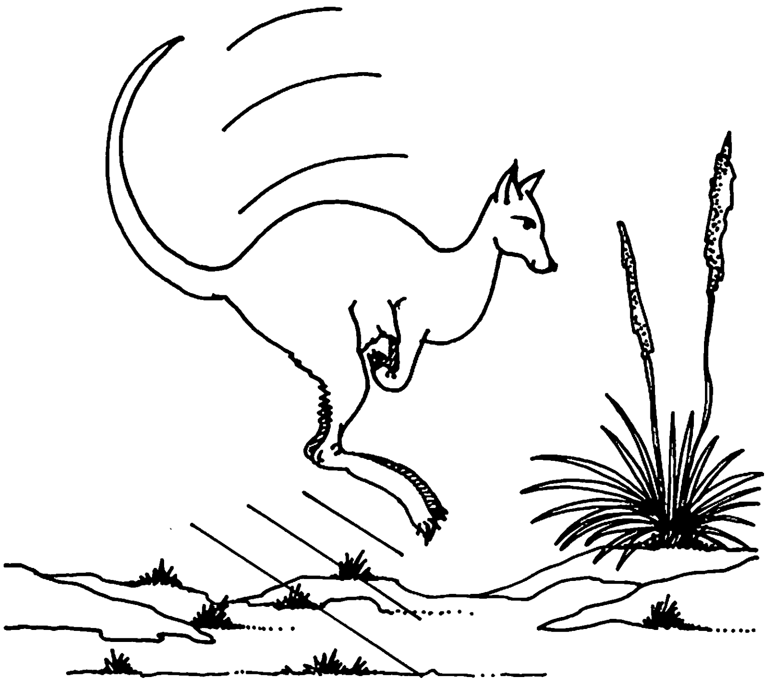
```
10 REM  ** HAPPY BIRTHDAY **
20 REM
30 REM  ** DAMON PILLINGER **
40 REM
50 REM  BASED ON 'DAY OF THE WEEK'
60 REM  BY MARK CHARLTON, IN
70 REM  "THE GATEWAY GUIDE..."
80 DAY$="SUNMONTUEWEDTHUFRISAT"
90 FOR JJ=1 TO 30
100 PRINT
110 NEXT JJ
120 PRINT "PLEASE ENTER THE DAY (AS"
130 PRINT "  A NUMBER, EG 30) ON"
140 INPUT "  WHICH YOU WERE BORN ":DAY
150 IF (DAY<1)+(DAY>31)THEN 120
160 PRINT
170 INPUT "WHAT MONTH ":MO
180 IF (MO<1)+(MO>12)THEN 170
190 PRINT :::
200 INPUT "WHAT YEAR   XXXX ":YEAR
210 REM  IF YEAR>100 THEN 210
220 REM  YEAR=YEAR+1900
230 CALL CLEAR
240 COY=0
250 IF MO>2 THEN 270
260 COY=1
270 LA=YEAR-COY
280 OA=MO+(12*COY)
290 PA=INT(LA/100)
300 DAT=INT((5*LA)/4)+D-1+INT
          (13*(OA+1)/5)-P+INT(P/4)
310 DAT1=(DAT-7*INT(DAT/7))*3+1
320 PRINT ,,,,,,
330 PRINT "THE DATE:"
340 PRINT "          ";DAY;"/";MO;"/";YEAR
350 PRINT
360 PRINT "IS A ";SEG$(DAY$,DAT1,3)
370 PRINT ::::::
380 FOR JJ=1 TO 421
390 NEXT JJ
400 GOTO 90
```

DOWNUNDER

Kevin Burfitt uses his computer to recreate some of the Australian landscape. You have to move your kangaroo down the screen to the door you can see near the bottom. As you bounce down you have to make sure you don't run into the river, the dingo, the platypus or the wombat.

You have extremely limited time to get to the door. You can increase your strength, and thus add to your chances of getting to the door, by eating the Vegemite that appears on the screen. (Vegemite, by the way, is a black paste which Aussies eat on nearly everything. It is extremely good for kangaroos.)

Once you get to the door, a new screen will appear; this one will be more difficult to navigate through.



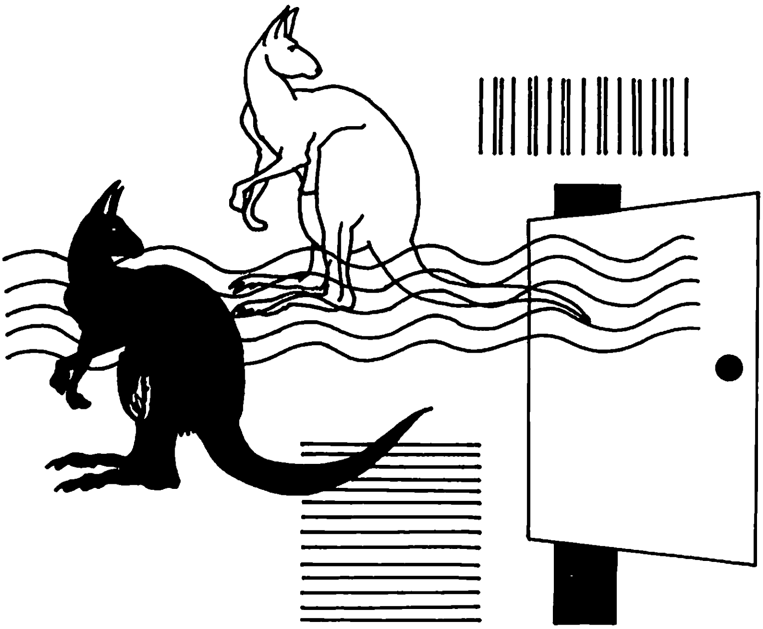
```
10 REM  *************
20 REM  * DOWNUNDER *
30 REM  *************
40 REM  BY Kevin Burfitt
50 REM
60 CALL CLEAR
70 PRINT "DOWNUNDER"::"MOVE YOUR KANGAROO
   WITH THE ARROW KEYS"::"TRY TO REACH TH
   E DOOR (";CHR$(30);")"
80 PRINT "AT THE BOTTOM OF THE SCREEN.DONT
    RUN INTO THE RIVER (f),DINGO (p),PLAT
    YPUS (h)":"OR THE WOMBAT (e)"
90 PRINT ::"GET POINTS FOR HITTING THE
 MAGIC ITEMS (ijklm),BUT MAKESURE YOU DON
 T RUN OUT OF":"TIME BY EATING";
100 PRINT " VEGEMITE (g)"::"GOOD LUCK"::
110 REM
120 RANDOMIZE
130 DEF BIN(X)=((L/2^X)<>INT(L/2^X))*
    (((L/2^X)-INT(L/2^X))>=0.5)
140 REM
150 REM  SETUP
160 LIVES=3
170 SCORE=0
180 CALL CHAR(100,"060704070E1E24CF")
190 CALL CHAR(101,"00037FFEFEFC486C")
200 CALL CHAR(99,"183C7E7E3C18183C")
210 CALL CHAR(102,"04E318C6318C6318")
220 CALL CHAR(103,"007E7E66667E3C18")
230 CALL CHAR(104,"0000003FFF123600")
240 CALL CHAR(105,"00424218184242")
250 CALL CHAR(106,"00544554455445")
260 CALL CHAR(107,"007E425A5A427E")
270 CALL CHAR(108,"0042424242422418")
280 CALL CHAR(109,"001818FFFF1818")
290 CALL CHAR(112,"0000027FFC78486C")
300 INPUT "STARTING LEVEL(1..5)?":BL
310 IF (BL>5)+(BL<1)THEN 300
320 L=BL
330 GOSUB 1730
340 CALL HCHAR(24,1,32,32)
350 X=17
360 CALL HCHAR(24,17,100)
370 GOSUB 1300
380 PRINT :
390 FOR I=2 TO 23
400 IF I=R THEN 550
410 CALL HCHAR(24,1,32,32)
```

```
420 IF I=D THEN 470
430 FOR J=1 TO -L*(L<11)-10*(L>10)
440 CALL HCHAR(24,RND*31+1,99)
450 NEXT J
460 GOTO 570
470 D=INT(RND*29)+1
480 CALL HCHAR(24,1,102,32)
490 CALL HCHAR(24,D,32,2)
500 D=24-INT(RND*I)
510 CALL HCHAR(D,1,102,32)
520 CALL HCHAR(D,D,32,2)
530 D=1
540 GOTO 570
550 CALL HCHAR(24,1,102,32)
560 CALL HCHAR(24,RND*28+1,32,3)
570 PRINT :
580 NEXT I
590 CALL HCHAR(RND*23+1,RND*31+1,32+71*F)
600 CALL HCHAR(24,1,32,32)
610 CALL HCHAR(24,RND*31+1,30)
620 IF W=0 THEN 650
630 WY=INT(RND*23)+1
640 WX=1
650 IF P=0 THEN 680
660 PY=INT(RND*23)+1
670 PX=1
680 GOSUB 1490
690 CALL KEY(1,K,S)
700 XX=X+(K=2)-(K=3)
710 YY=Y+(K=5)-(K+1=1)
720 IF (XX<1)+(YY<1)+(XX>32)+
        (YY>24)+(S=0)THEN 800
730 CALL GCHAR(YY,XX,HIT)
740 IF (HIT=32)+(HIT=100)THEN 770
750 IF HIT=103 THEN 1150
760 IF HIT=99 THEN 800 ELSE 1190
770 CALL HCHAR(Y,X,32)
780 Y=YY
790 X=XX
800 CALL HCHAR(Y,X,100)
810 IF W=0 THEN 870
820 CALL HCHAR(WY,WX,32)
830 WX=WX+1+32*(WX=32)
840 CALL GCHAR(WY,WX,HIT)
850 IF HIT=100 THEN 1080
860 CALL HCHAR(WY,WX,101)
870 IF P=0 THEN 930
880 CALL HCHAR(PY,PX,32)
890 PX=PX-1-32*(PX=1)
```

```
900 CALL GCHAR(PY,PX,HIT)
910 IF HIT=100 THEN 1080
920 CALL HCHAR(PY,PX,104)
930 IF DD<>1 THEN 970
940 DI=DX
950 DJ=DY
960 GOSUB 1800
970 TIME=TIME-1
980 IF TIME<>50 THEN 1000
990 DD=1
1000 DD=DD-1-(DD<2)
1010 IF TIME>20 THEN 690
1020 CALL SOUND(-100,440,0,441,2,442,4)
1030 IF TIME>1 THEN 690
1040 REM
1050 CALL SOUND(1000,440,0,441,2,442,4)
1060 PRINT "NO VEGEMITE"
1070 PRINT "No VEGEMITE !"
1080 LIVES=LIVES-1
1090 X=17
1100 TIME=200
1110 Y=1
1120 PRINT :"NEXT KANGAROO":
1130 IF LIVES>-1 THEN 370
1140 GOTO 1620
1150 CALL SOUND(100,660,0,880,2,1100,4)
1160 SCORE=SCORE+5
1170 TIME=TIME+20
1180 GOTO 770
1190 IF HIT=30 THEN 1220
1200 IF HIT>104 THEN 1560
1210 GOTO 1080
1220 SCORE=SCORE+10
1230 L=L+1
1240 TIME=TIME+10+(140-TIME)*F
1250 CALL HCHAR(Y,X,32)
1260 Y=1
1270 X=XX
1280 CALL HCHAR(YY,XX,100)
1290 GOTO 370
1300 F=BIN(1)
1310 TIME=-150*(L=BL)+TIME
1320 DX=1
1330 DY=24
1340 W=BIN(2)
1350 R=INT(RND*23)*BIN(3)+1
1360 P=BIN(4)
1370 D=BIN(5)
1380 DD=D
```

```
1390 CALL CHAR(32,"55AA55AA55AA55AA")
1400 FG=13-2*D
1410 BG=3+9*D
1420 CALL COLOR(1,FG,BG)
1430 FOR I=2 TO 16
1440 CALL COLOR(I,2,BG)
1450 NEXT I
1460 D=INT(RND*22)*D+1
1470 Y=1
1480 RETURN
1490 REM  SOUND
1500 N=1-(L<10)-(L<5)
1510 FOR I=N TO 5
1520 CALL HCHAR(RND*23+1,RND*31+1,104+I)
1530 NEXT I
1540 REM  CALL SOUND()
1550 RETURN
1560 SCORE=SCORE+110-HIT
1570 CALL HCHAR(YY,XX,48+110-HIT)
1580 CALL SOUND(100,440,1,441,2,442,3)
1590 DD=10+10*(DD=0)
1600 CALL SOUND(80,660,1,661,2,662,3)
1610 GOTO 770
1620 DATA 71,97,109,101,32,111,118,101,114
1630 RESTORE 1620
1640 FOR I=1 TO 9
1650 READ A
1660 CALL HCHAR(9,10+I,A)
1670 NEXT I
1680 PRINT :::::"YOUR SCORE ->";SCORE:::
1690 PRINT "DO YOU WISH TO CONTINUE
      (CON)OR PLAY A NEW GAME(NEW)"
1700 INPUT "CON or NEW ?":A$
1710 IF A$="CON" THEN 32767
1720 GOTO 32767
1730 DATA DOWNUNDER TUNE
1740 RESTORE 1730
1750 FOR I=1 TO []
1760 READ D,N,L
1770 CALL SOUND(D,N,L)
1780 NEXT I
1790 RETURN
1800 DI=DX+(X<DX)-(X>DX)
1810 CALL GCHAR(DY,DI,HIT)
1820 IF HIT=100 THEN 1080
1830 IF (HIT<>102)*(HIT<>30)*
         (HIT<>112)THEN 1900
1840 DI=DX
1850 DJ=DY+(Y<DY)-(Y>DY)
```

```
1860 CALL GCHAR(DJ,DX,HIT)
1870 IF HIT=100 THEN 1080
1880 IF (HIT<>102)*(HIT<>30)*
          (HIT<>112)THEN 1900
1890 DJ=DY
1900 CALL HCHAR(DY,DX,32)
1910 DX=DI
1920 DY=DJ
1930 CALL HCHAR(DY,DX,112)
1940 RETURN
```



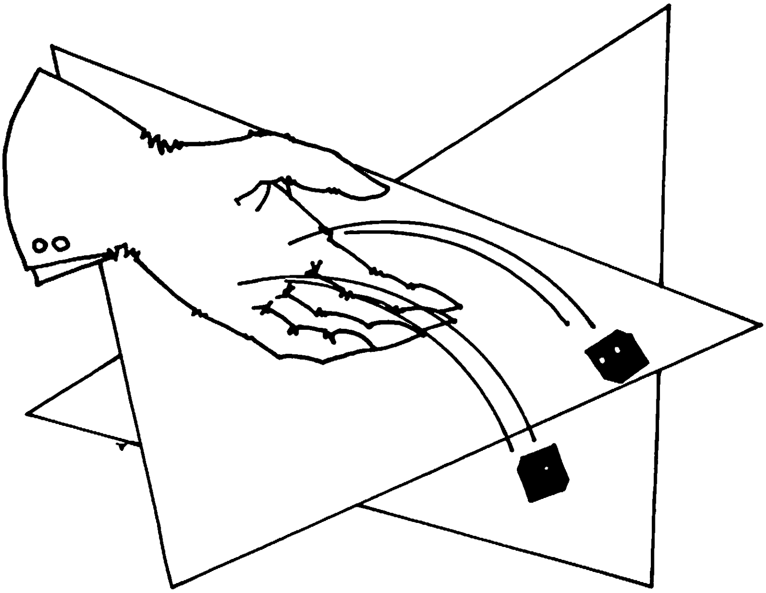
TEXAS TENBY

Texas Tenby is a gambling game played with dice. Written by Tim Hartnell, it is based on the game of Craps. It is very simple to play (but not so easy to win). There are 10 rounds to each game.

You roll two dice, and add their totals. The game ends on the first roll if the dice total 7 or 11. These totals, when gained on the first roll, are called 'a natural'; if you roll them you win.

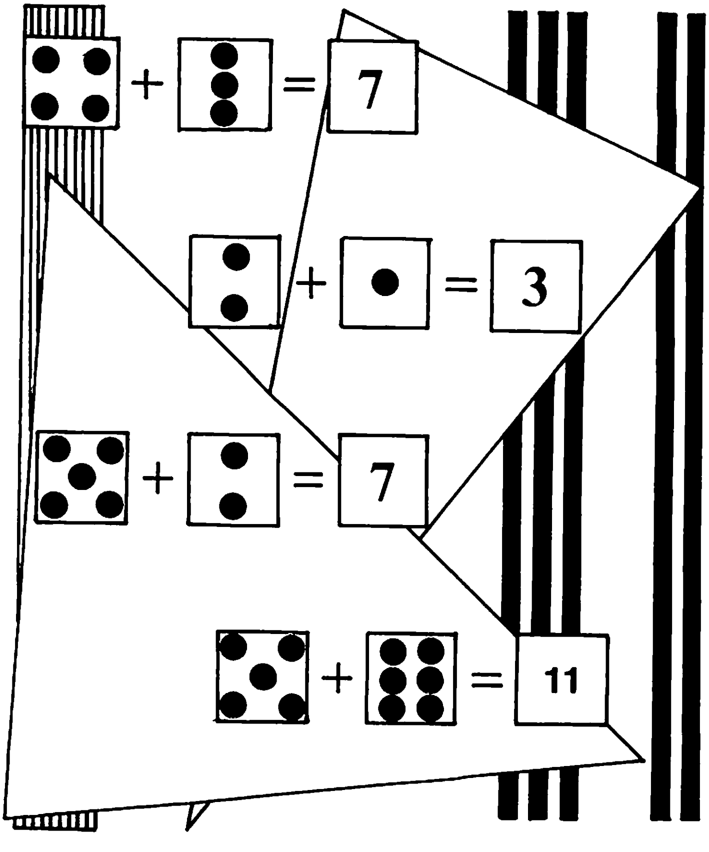
If you roll 2, 3 or 12 on the first roll ('craps') you lose. Any other total becomes your 'point'. To win, you need to roll your point again, *before* rolling a 7.

You will find that the TI does most of the work for you, rolling the dice, checking your wins and losses, and keeping track of things. The best way to play it is with a friend; take it in turns to 'roll the dice', with rolls 1, 3, 5, 7 and 9 for you, and the other rolls for your friend.



```
10 REM  TEXAS TENBY
20 CALL CLEAR
30 B$="You rolled"
40 RANDOMIZE
50 G=0
60 W=0
70 L=0
80 G=G+1
90 IF G=11 THEN 490
100 PRINT "---------------------------"
110 PRINT "--------NEW GAME-----------"
120 PRINT "---------------------------"::
130 PRINT "WINS:";W,"LOSSES:";L
140 PRINT ::"GAME NUMBER";G
150 GOSUB 270
160 IF (A=7)+(A=11)<0 THEN 350
170 IF (A=2)+(A=3)+(A=12)<0 THEN 380
180 P=A
190 PRINT :"Your point is";P
200 GOSUB 270
210 IF A=P THEN 350
220 IF A=7 THEN 380
230 FOR T=1 TO 300
240 NEXT T
250 PRINT :"^^^^^^^^^^^^^^^^^^^^^^^^^^^^":
260 GOTO 190
270 A=INT(RND*6+RND*6+2)
280 IF A>12 THEN 270
290 FOR T=1 TO 100
300 NEXT T
310 GOSUB 450
320 GOSUB 450
330 PRINT ::,B$;A
340 RETURN
350 PRINT :"You win!"
360 W=W+1
370 GOTO 400
380 PRINT :"You lose!"
390 L=L+1
400 FOR T=1 TO 400
410 NEXT T
420 GOSUB 450
430 GOSUB 450
440 GOTO 80
450 REM  SOUND ROUTINE
460 NOTE=INT(RND*200)+178
470 CALL SOUND((RND*1000+100),NOTE,0,
   (NOTE+220),2)
```

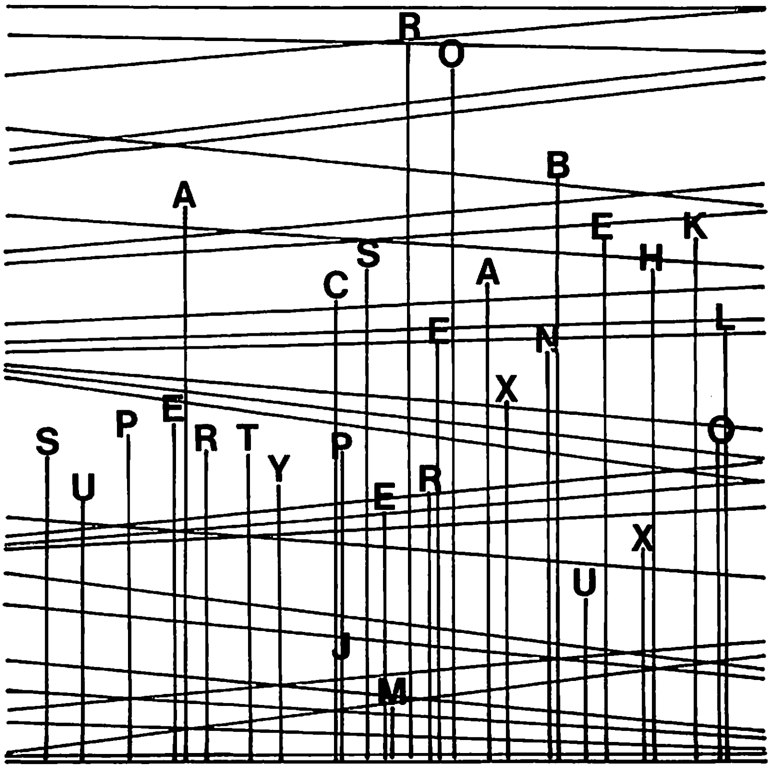
```
480 RETURN
490 REM  END OF GAME
500 PRINT ::"THAT'S THE END OF THE GAME"
510 PRINT :::"LET'S SEE HOW YOU WENT..."
520 FOR T=1 TO 4
530 GOSUB 450
540 NEXT T
550 IF W>L THEN 590
560 IF L<W THEN 610
570 PRINT ::"THE WINS EQUALLED THE LOSSES"
580 END
590 PRINT :"YOU WON BY";W;"TO";L
600 END
610 PRINT :"YOU LOST BY";L;"TO";W
620 END
```



SUPER TYPER

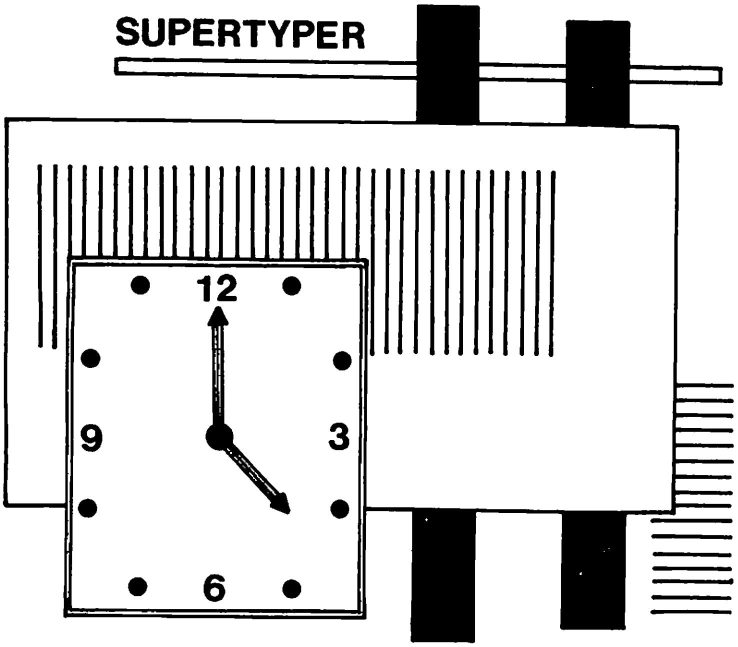
In this dynamic game from K. Burfitt, you and your typing fingers are set against a relentless clock, as you attempt to stop randomly-chosen letters of the alphabet from crawling to the top of the screen.

Letters will appear at the bottom of the screen — one by one — and then start making their way towards the top. You have to press the relevant letters on your keyboard before the letters reach the top of the screen. You will discover that the letters move more and more quickly as your score gets higher. There is a high-score feature.



```
10 REM  SUPER TYPER MARK II
20 REM
30 REM  K BURFITT
40 REM
50 HS$="SUPER TYPER MARK "&CHR$(127)
60 HSCORE=400
70 CALL CHAR(127,"FF42424242424242FF")
80 CALL CLEAR
90 RANDOMIZE
100 GOTO 540
110 FOR I=0 TO 9
120 A(I)=INT(RND*26)+65
130 B(I)=23
140 NEXT I
150 FOR I=0 TO 9
160 CALL HCHAR(B(I),I*2+5,A(I))
170 NEXT I
180 CALL KEY(0,K,S)
190 IF S<1 THEN 320
200 FOR I=0 TO 9
210 IF A(I)<>K THEN 310
220 CALL VCHAR(1,I*2+5,32,24)
230 SCORE=SCORE+5
240 CALL SOUND(-200,-6,0)
250 A(I)=INT(RND*26)+65
260 B(I)=23
270 FOR II=1 TO LEN(STR$(SCORE))
280 CALL HCHAR(8+II,2,ASC(SEG$
                    (STR$(SCORE),II,1)))
290 NEXT II
300 CALL HCHAR(B(I),I*2+5,A(I))
310 NEXT I
320 X=INT(RND*10)
330 B(X)=B(X)-1-SCORE/100
340 IF B(X)<0.5 THEN 410
350 CALL HCHAR(B(X)+1+SCORE/100,X*2+5,32)
360 FOR I=B(X)+1+SCORE/100 TO B(X)STEP -1
370 CALL HCHAR(I+1,X*2+5,32)
380 CALL HCHAR(I,X*2+5,A(X))
390 NEXT I
400 GOTO 180
410 PRINT "YOU GOT A SCORE OF ";SCORE;
420 FOR I=1 TO 1500
430 NEXT I
440 CALL CLEAR
450 IF SCORE<HSCORE THEN 540
460 PRINT "YOU HAVE TOP SCORE "
470 PRINT "THE PREVIOUS BEST WAS";
    HSCORE;"BY     ";HS$
```

```
480 INPUT "WHAT IS YOUR NAME ?":HS$
490 IF LEN(HS$)<25 THEN 520
500 PRINT "ONLY 24 CHARACTERS LONG,
    NOT";LEN(HS$);" CHARACTERS PLEASE"
510 GOTO 480
520 HSCORE=SCORE
530 CALL CLEAR
540 FOR I=1 TO LEN(HS$)
550 CALL HCHAR(I,26,ASC(SEG$(HS$,I,1)))
560 NEXT I
570 S$=STR$(HSCORE)
580 FOR I=1 TO LEN(S$)
590 CALL HCHAR(I,27,ASC(SEG$(S$,I,1)))
600 NEXT I
610 Q$="PRESS A KEY TO START"
620 FOR I=1 TO LEN(Q$)
630 CALL HCHAR(4,I+2,ASC(SEG$(Q$,I,1)))
640 NEXT I
650 CALL KEY(0,K,S)
660 IF S=0 THEN 650
670 CALL HCHAR(4,2,32,22)
680 SCORE=0
690 GOTO 110
```



TI FASTERMIND

Now you can challenge your computer to the well-known game often called 'Codebreaker'. The computer picks a four-digit number (with no digits repeated); you have to try and work out what the number is.

You have 10 attempts to do this. After each guess, the computer will respond with a score in 'blacks' and 'whites'. The blacks represent correct digits in correct positions within the code you entered; whites are correct digits, but not in the right position. This game is based on a program by Toni Baker.

```
10 REM  TI FASTERMIND
20 RANDOMIZE
30 CALL CLEAR
40 C(1)=INT(RND*9)+1
50 Z=2
60 C(Z)=INT(RND*9)+1
70 J=1
80 IF C(J)=C(Z)THEN 50
90 IF J=Z-1 THEN 120
100 J=J+1
110 GOTO 80
120 IF Z=4 THEN 150
130 Z=Z+1
```

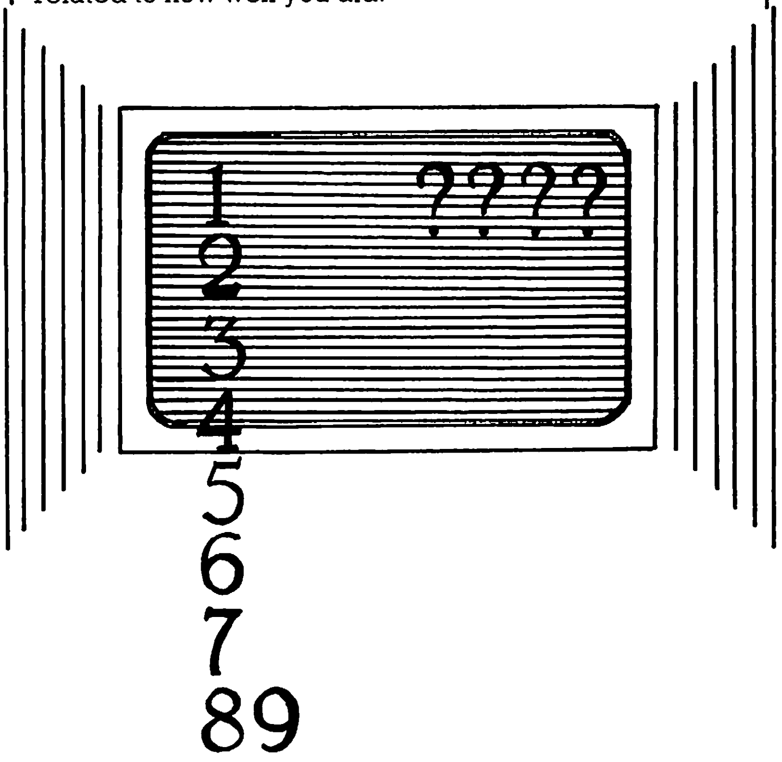
```
140 GOTO 60
150 FOR CHANCES=1 TO 10
160 INPUT A
170 IF A<1000 THEN 160
180 IF A>9999 THEN 160
190 A1=A
200 FOR Z=1 TO 4
210 G(Z)=A-10*(INT(A/10))
220 A=INT(A/10)
230 NEXT Z
240 B=0
250 FOR Z=1 TO 4
260 W=0
270 IF C(Z)<>G(Z)THEN 300
280 B=B+1
290 G(Z)=0
300 NEXT Z
310 FOR Z=1 TO 4
320 IF G(Z)=0 THEN 370
330 FOR J=1 TO 4
340 IF C(Z)<>G(J)THEN 360
350 W=W+1
360 NEXT J
370 NEXT Z
380 PRINT B;"black";
390 IF B=1 THEN 470
400 IF B<>1 THEN 490
410 PRINT "  ";W;"white";
420 IF W<>1 THEN 510
430 PRINT
440 IF B=4 THEN 530
450 NEXT CHANCES
460 GOTO 540
470 PRINT " ";
480 GOTO 410
490 PRINT "s";
500 GOTO 410
510 PRINT "s";
520 GOTO 430
530 PRINT ::"YOU GUESSED IT!"
540 PRINT ::"The code was ";
550 FOR Z=4 TO 1 STEP -1
560 PRINT STR$(C(Z));
570 NEXT Z
580 FOR Z=1 TO 3000
590 NEXT Z
600 FOR Z=1 TO 30
610 PRINT
620 NEXT Z
```

SIMON

This is a memory-test program, in which you have to try and repeat a constantly-growing string of digits.

The game starts quietly, with one digit. The screen will clear, and you have to enter the same number. If you are right, the computer will reprint the first number, and follow it with a second one. Once the screen has cleared again, you have to enter both numbers, in the right order, pressing ENTER after each digit.

This harrowing process will continue until you either manage to get 10 digits in a row right, or you make a mistake. You will be given a score at the end of the game related to how well you did.



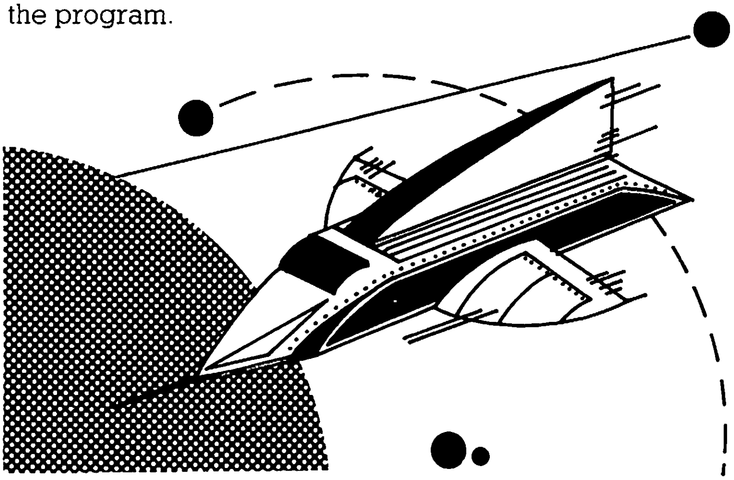
```
10 REM  SIMON
20 CALL CLEAR
30 CALL SCREEN(16)
40 CALL COLOR(2,7,16)
50 FOR T=1 TO 10
60 A(T)=INT(RND*10)+1
70 PRINT "*******************":
80 NEXT T
90 FOR Y=1 TO 500
100 NEXT Y
110 FOR T=1 TO 10
120 CALL CLEAR
130 FOR Z=1 TO T
140 PRINT A(Z);
150 RT=500+110*A(Z)
160 CALL SOUND(50,RT,0)
170 NEXT Z
180 PRINT
190 FOR Y=1 TO 100+4*T
200 NEXT Y
210 CALL CLEAR
220 FOR Z=1 TO T
230 INPUT K
240 RT=500+110*K
250 CALL SOUND(-100,RT,0)
260 CALL CLEAR
270 IF K<>A(Z)THEN 380
280 PRINT K;
290 NEXT Z
300 FOR Y=1 TO 500
310 NEXT Y
320 CALL CLEAR
330 FOR Y=1 TO 200
340 NEXT Y
350 NEXT T
360 PRINT "YOU ARE THE CHAMP!"
370 END
380 PRINT :::
390 PRINT "THE NUMBER WAS ";
400 FOR Z=1 TO T
410 PRINT A(Z);
420 RT=500+100*A(Z)
430 CALL SOUND(100,RT,0)
440 NEXT Z
450 PRINT :
460 PRINT :
470 PRINT "YOU SCORED";79*T
```

DIAMOND FEVER

Here is another great game — which makes effective use of TI graphics — written by Kevin Burfitt. You are alone on an asteroid in deepest space when the program begins. While scanning the asteroid, you notice it has a high metal-to-rock ratio. As a miner, you realise that you will be extremely wealthy if you mine the asteroid for its ore.

But your dreams of easy riches are shattered when you see that there is an evil Vogon spacecraft within the asteroid. Your task is to move from one side of the asteroid (you start on the left) to the exit on the other side, before the Vogon gets you.

You have to drill your way out of the mine. Metal takes a long time to work through, and diamond is practically impossible to penetrate. When the game is over, the location of the metal and diamonds will be shown to you; you do not, however, know where they are when you are trying to elude the Vogon. Full instructions are given within the program.



```
10 REM  DIAMOND FEVER
20 CALL CLEAR
30 INPUT "DO YOU WANT
         INSTRUCTIONS ?":HUH$
40 IF SEG$(HUH$,1,1)<>"Y" THEN 290
50 PRINT "         DIAMOND FEVER"
60 PRINT "           ~~~~~~~"
70 PRINT :"        written by":
80 PRINT :"       Kevin Burfitt"
90 PRINT "            1983"
100 PRINT :::::::::
110 FOR I=1 TO 1000
120 NEXT I
130 CALL CLEAR
140 PRINT "        You are alone, on an
     asteroid in deepest space.  While
     scanning the rock"
150 PRINT " you notice it contains an
    abnormaly  high  amount  of metals,
    so you venture"
160 PRINT "down to see if you can mine
   any   WHEN   SUDDENLY   you realise
   that you are not "
170 PRINT "inside an asteroid but  an
      evil VOGON spacecraft !!"::
180 PRINT " How can you escape from the
     maze of rock and rubble?  Fate was
     on your side, you  got a look at the
     diamond"
190 PRINT "seam in the rock,but can you
     reach the exit before the "
200 PRINT :"   Vogon gets you ????"::
210 PRINT :"PRESS ANY KEY TO CONTINUE"
220 CALL KEY(0,KEY,STATUS)
230 IF STATUS THEN 240 ELSE 220
240 CALL SOUND(100,440,0)
250 PRINT ::"level 1 is easiest > level
 5 for the champions "::"you have your tr
 usty blaster with"
260 PRINT "you and it's running out of
 steam (uraniam actually) but it will c
 ut through rock"::"N.B. YOU ARE RUNNING
 OUT OF"
270 PRINT :"  gasp'  AIR  gasp'"::
280 PRINT "Use the arrow keys to move":
"Your blaster will  'bleep'  when you try
  to cut through rock"
290 HIGHEST_SCORER$="HIGH SCORE"
```

```
300 HIGH_SCORE=50
310 CALL CHAR(132,"995A3C3C3C3C2424")
320 DIAMOND=3
330 ROCK=1
340 SCORE=0
350 CALL CHAR(136,"55AA55AA55AA55AA")
360 DIM A(19,19)
370 FOR I=1 TO 19
380 FOR J=1 TO 19
390 A(I,J)=0
400 NEXT J
410 NEXT I
420 RANDOMIZE
430 INPUT "LEVEL OF DIFFICULTY ?":L
440 SC1=0
450 CALL CLEAR
460 IF (L<0)+(L>5)THEN 430
470 LEVEL=L
480 CALL COLOR(10,3,3)
490 CALL COLOR(11,2,2)
500 CALL COLOR(13,2,3)
510 CALL COLOR(14,2,16)
520 CALL CHAR(133,"0018245A18240000")
530 CALL CLEAR
540 CALL SCREEN(L+4)
550 AX=10
560 PRINT ::::::"  DIAMOND SEAM "::
570 AY=INT(RND*4)+2
580 FOR N=1 TO 30+(LEVEL)
590 DIR=INT(RND*5)+1
600 AX=AX+(DIR=1)-(DIR=3)
610 AY=AY-(DIR=2)-(DIR=4)+(DIR=5)
620 AX=AX-(AX<1)+(AX>19)
630 AY=AY-(AY<1)+(AY>19)
640 CALL HCHAR(AX,AY,136)
650 A(AX,AY)=DIAMOND
660 NEXT N
670 FOR N=1 TO 80+4*LEVEL
680 Y=INT(RND*19)+1
690 X=INT(RND*18)+2
700 IF A(Y,X)=DIAMOND THEN 760
710 IF A(Y,X)=ROCK THEN 770
720 A(Y,X)=0
730 IF ((Y=9)+(Y=10))*((X=18)
              +(X=19))THEN 770
740 A(Y,X)=ROCK
750 GOTO 770
760 A(Y,X)=0
770 NEXT N
```

```
780 CALL CLEAR
790 Y=10
800 X=1
810 PRINT "LEVEL -> ";LEVEL:
820 R=Y
830 FOR I=1 TO LEN(STR$(SCORE))
840 CALL HCHAR(2,22+I,ASC(SEG$
            (STR$(SCORE),I,I)))
850 NEXT I
860 V=19
870 FOR I=1 TO LEN(STR$(HIGH_SCORE))
880 CALL HCHAR(6,22+I,ASC(SEG$(STR$
                    (HIGH_SCORE),I,I)))
890 NEXT I
900 C=0
910 FOR I=1 TO LEN(HIGHEST_SCORER$)
920 CALL HCHAR(5,22+I,ASC(SEG$
                (HIGHEST_SCORER$,I,I)))
930 NEXT I
940 CALL HCHAR(21,1,112,21)
950 CALL HCHAR(1,1,112,21)
960 CALL VCHAR(1,21,112,21)
970 CALL VCHAR(1,1,112,21)
980 FOR I=2 TO 20
990 CALL HCHAR(I,2,136,19)
1000 NEXT I
1010 CALL HCHAR(11,21,32)
1020 FOR N=1 TO 8-L
1030 C=C+1
1040 IF C>300+L*100 THEN 1910
1050 IF 50*LEVEL-C>0 THEN 1110
1060 TEST=TEST=0
1070 IF TEST=0 THEN 1090
1080 CALL SCREEN(16)
1090 IF TEST=-1 THEN 1110
1100 CALL SCREEN(L+4)
1110 IF C<20+L*100 THEN 1130
1120 CALL SOUND(-200,770,0,769,0,771,0)
1130 Z=X
1140 T=Y
1150 CALL KEY(0,KEY,STATUS)
1160 Y=Y+(KEY=ASC("E"))-(KEY=ASC("X"))
1170 X=X+(KEY=ASC("S"))-(KEY=ASC("D"))
1180 IF (Y=10)*(X=20)THEN 1560
1190 Y=Y-(Y<1)+(Y>19)
1200 X=X-(X<1)+(X>19)
1210 IF (R=Y)*(V=X)THEN 1910
1220 IF A(Y,X)=0 THEN 1330
1230 A(Y,X)=A(Y,X)-0.1
```

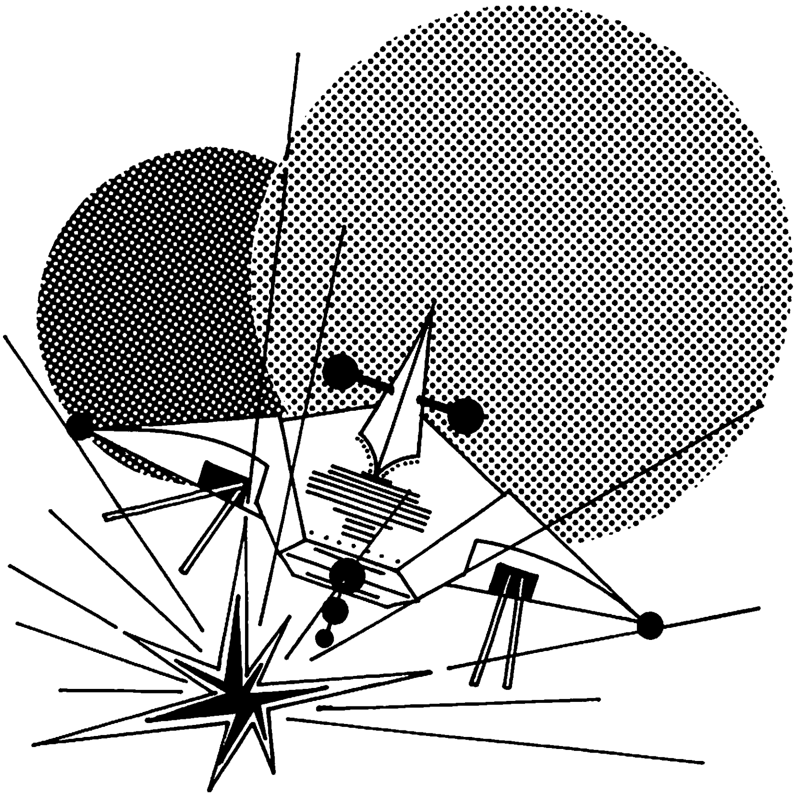
```
1240 IF A(Y,X)>0 THEN 1290
1250 SCORE=SCORE+LEVEL
1260 FOR I=1 TO LEN(STR$(SCORE))
1270 CALL HCHAR(2,22+I,ASC
            (SEG$(STR$(SCORE),I,I)))
1280 NEXT I
1290 X=Z
1300 Y=T
1310 CALL SOUND(-50,880,0)
1320 GOTO 1360
1330 REM
1340 CALL HCHAR(T+1,Z+1,111)
1350 CALL HCHAR(Y+1,X+1,132)
1360 REM
1370 NEXT N
1380 E=R
1390 S=V
1400 V=V-(V<X)+(V>X)
1410 IF A(R,V)=0 THEN 1440
1420 V=S
1430 GOTO 1470
1440 CALL HCHAR(E+1,S+1,111)
1450 CALL HCHAR(R+1,V+1,133)
1460 S=V
1470 R=R-(R<Y)+(R>Y)
1480 IF A(R,V)=0 THEN 1520
1490 R=E
1500 V=S
1510 GOTO 1540
1520 CALL HCHAR(E+1,S+1,111)
1530 CALL HCHAR(R+1,V+1,133)
1540 IF (R=Y)*(V=X)THEN 1910
1550 GOTO 1020
1560 CHECK=50*LEVEL-C
1570 IF CHECK<0 THEN 1640
1580 IF CHECK<300 THEN 1600
1590 CHECK=300
1600 SCORE=SCORE+CHECK+50
1610 IF SCORE>0 THEN 1640
1620 SCORE=0
1630 SC1=0
1640 GOSUB 2130
1650 SC2=SCORE-SC1
1660 SC1=SCORE
1670 PRINT "YOU GOT ";SC2;" THAT TIME";
1680 GOSUB 2010
1690 LEVEL=LEVEL+1
1700 L=L+1
1710 FOR I=1 TO 19
1720 FOR J=1 TO 19
```

```
1730 A(I,J)=0
1740 NEXT J
1750 NEXT I
1760 IF L<8 THEN 530
1770 L=1
1780 DIAMOND=DIAMOND+0.5
1790 ROCK=ROCK+0.2
1800 GOTO 530
1810 IF SCORE>HIGH_SCORE THEN 1850
1820 PRINT "THE HIGH SCORE IS ";HIGH_SCORE;
        " AND YOU GOT ";SCORE
1830 PRINT ::::::
1840 GOTO 310
1850 HIGH_SCORE=SCORE
1860 PRINT "THE HIGH SCORE IS ";
     HIGH_SCORE;" AND YOU GOT IT"
1870 INPUT "WHAT IS YOUR NAME ?"
     :HIGHEST_SCORER$
1880 IF LEN(HIGHEST_SCORER$)<11 THEN 310
1890 PRINT "NOT THAT LONG"
1900 GOTO 1870
1910 GOSUB 1960
1920 FOR I=1 TO 1000
1930 NEXT I
1940 CALL CLEAR
1950 GOTO 1810
1960 FOR I=1 TO 3
1970 CALL SOUND(100,220,0,219,0,218,0)
1980 CALL SOUND(1,110,30)
1990 NEXT I
2000 CALL SOUND(1500,185,0,184,0,183,0)
2010 FOR I=1 TO 19
2020 FOR J=1 TO 19
2030 IF A(I,J)=0 THEN 2090
2040 IF A(I,J)<ROCK+0.1 THEN 2070
2050 CALL HCHAR(I+1,J+1,112)
2060 GOTO 2100
2070 CALL HCHAR(I+1,J+1,136)
2080 GOTO 2100
2090 CALL HCHAR(I+1,J+1,111)
2100 NEXT J
2110 NEXT I
2120 RETURN
2130 FOR I=135 TO 216 STEP 10
2140 CALL SOUND(100,I,0,I+1,0,I+2,0)
2150 NEXT I
2160 CALL SOUND(200,40000,30)
2170 CALL SOUND(300,110,0,111,0,112,0)
2180 RETURN
```

FINAL FRONTIER

You won't have much time to catch your breath in this action game from Damon Pillinger.

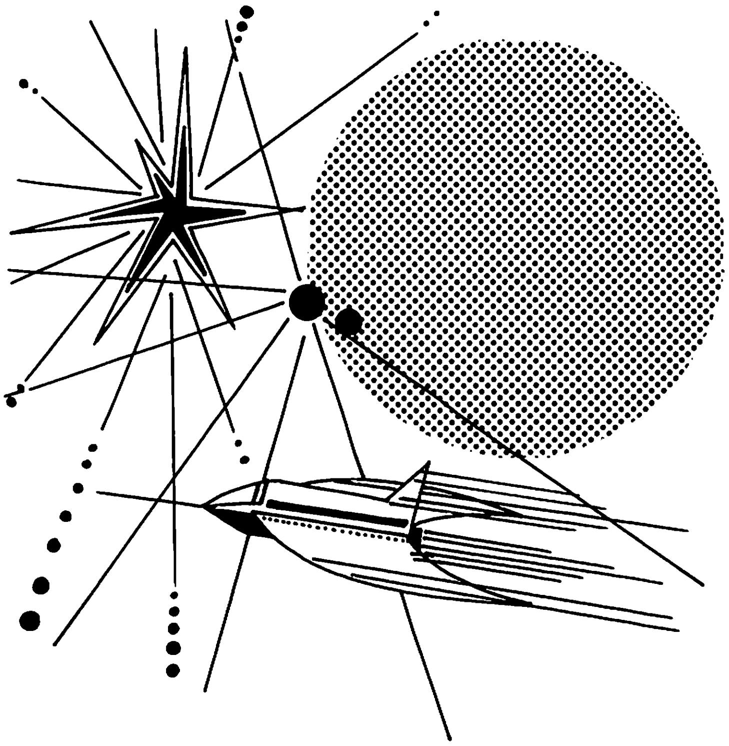
Once again, it is you who are the last line of Earth's defence. You use the up and down arrow keys to position your craft, and the space bar to fire. The game becomes more and more difficult as it progresses.



```
10 REM   ** FINAL FRONTIER **
20 REM
30 REM  ** D&D PROGRAMMING *
40 REM
50 REM   ** OF GAMETRONICS **
60 REM  **** MELBOURNE *****
70 REM
80 REM  **** WRITTEN BY ****
90 REM  ****** DAMON *******
100 REM
110 LEVEL=1
120 CALL SCREEN(2)
130 K=28
140 CALL CLEAR
150 LEVEL=LEVEL+3
160 HP=10
170 RANDOMIZE
180 DED=0
190 CALL SCREEN(2)
200 CALL COLOR(9,6,1)
210 CALL COLOR(8,3,3)
220 CALL COLOR(10,4,1)
230 CALL COLOR(3,7,1)
240 CALL COLOR(11,11,1)
250 CALL COLOR(2,14,1)
260 DIM P1(3)
270 P1(1)=10
280 P1(2)=15
290 P1(3)=22
300 CALL CHAR(49,"8151565616ABC566")
310 CALL CHAR(50,"3136380FED9BCA71")
320 CALL CHAR(51,"5554765616FE6154")
330 CALL CHAR(40,"0080C0F08F7C7FF0")
340 CALL CHAR(41,"00000000C030FF00")
350 A$="000000000F3F0100"
360 B$="003F7F8EFCF0F800"
370 CALL CHAR(96,A$)
380 CALL CHAR(97,B$)
390 CALL CHAR(90,"FFFFFFFFFFFFFFFF")
400 CALL CHAR(104,A$)
410 CALL CHAR(48,"00000000000000FF")
420 CALL CHAR(105,B$)
430 CALL CHAR(112,A$)
440 CALL CHAR(113,B$)
450 CALL HCHAR(1,1,90,32)
460 CALL HCHAR(22,1,90,32)
470 FOR T=K TO 1 STEP -(LEVEL/10)
480 FOR Q=1 TO INT(LEVEL/3.3333333)
490 IF P1(Q)=999 THEN 560
```

```
500 CALL HCHAR(P1(Q),T,32,4)
510 P1(Q)=P1(Q)+(INT((RND*2)-.91))
520 IF (P1(Q)>2)*(P1(Q)<22)THEN 540
530 P1(Q)=INT(RND*10)+3
540 CALL HCHAR(P1(Q),T,88+Q*8)
550 CALL HCHAR(P1(Q),T+1,89+Q*8)
560 NEXT Q
570 CALL KEY(3,X,Y)
580 IF Y=0 THEN 690
590 CALL HCHAR(HP,2,32,2)
600 IF (X=69)*(HP>2)THEN 620
610 GOTO 630
620 HP=HP-1
630 IF (X=88)*(HP<21)THEN 650
640 GOTO 660
650 HP=HP+1
660 CALL HCHAR(HP,2,40)
670 CALL HCHAR(HP,3,41)
680 IF X=32 THEN 720
690 REM  END OF YOUR MOVING LOOP
700 NEXT T
710 GOTO 980
720 CALL HCHAR(HP,4,48,28)
730 CALL SOUND(-70,-7,0,300,7)
740 FOR Y=1 TO INT(LEVEL/3.3)
750 IF P1(Y)=HP THEN 800
760 NEXT Y
770 CALL HCHAR(HP,4,32,28)
780 CALL HCHAR(HP,3,41)
790 GOTO 700
800 FOR A=1 TO 5
810 CALL HCHAR(P1(Y),T,49,2)
820 CALL HCHAR(P1(Y),T,50,2)
830 CALL HCHAR(P1(Y),T,51,2)
840 CALL SOUND(-200,-7,0)
850 NEXT A
860 CALL HCHAR(HP,4,32,28)
870 CALL SOUND(-300,-7,0)
880 DED=DED+1
890 SCORE=SCORE+(LEVEL-2)*15
900 IF DED<>INT(LEVEL/3.3)THEN 960
910 LEVEL=LEVEL+1
920 IF LEVEL<>14 THEN 950
930 K=K/1.5
940 LEVEL=4
950 GOTO 170
960 P1(Y)=999
970 GOTO 660
980 FOR H=1 TO 5
```

```
990 FOR T=1 TO 16
1000 CALL SCREEN(T)
1010 NEXT T
1020 NEXT H
1030 CALL COLOR(8,2,16)
1040 DIM X$(4)
1050 X$(1)="YOU HAVE LOST.."
1060 X$(2)="THE ALIENS HAVE WON..."
1070 X$(3)=" "
1080 X$(4)="YOUR SCORE IS "
1090 FOR Y=1 TO 4
1100 FOR T=1 TO LEN(X$(Y))
1110 CALL HCHAR(Y+1,T,ASC(SEG$(X$(Y),T,1)))
1120 NEXT T
1130 NEXT Y
1140 FOR AA=1 TO 300
1150 NEXT AA
1160 CALL CLEAR
1170 PRINT "SCORE=";SCORE:,,,,
     "LEVEL";LEVEL-3,,,,,,
```



FENCED IN

James Turner has contributed this hard-to-beat game, in which you have to get from the top left-hand corner of the screen to the bottom right-hand one, through a field of electric fences.

You are only allowed to touch five fences before you die (and the graphic result of dying is enough to tempt you to do it deliberately). Each time you touch a fence, you'll clear a few other fences away; this will make it easier to get to your goal.

In this game of strategy, there is no back-tracking (you can only move right, and down) so you must work out your moves in advance. You use the 'M' key to move down, and the 'L' key to move right.

```
10 REM  FENCED IN
20 CALL CLEAR
30 PRINT TAB(9);"FENCED IN!"::
40 PRINT TAB(13);"By"::
50 PRINT TAB(8);"James turner"::::::::
60 FOR I=1 TO 1000
70 NEXT I
80 CALL CLEAR
90 CALL CHAR(129,"00FF00FF00FF00FF")
100 CALL CHAR(130,"AAAAAAAAAAAAAAAA")
```

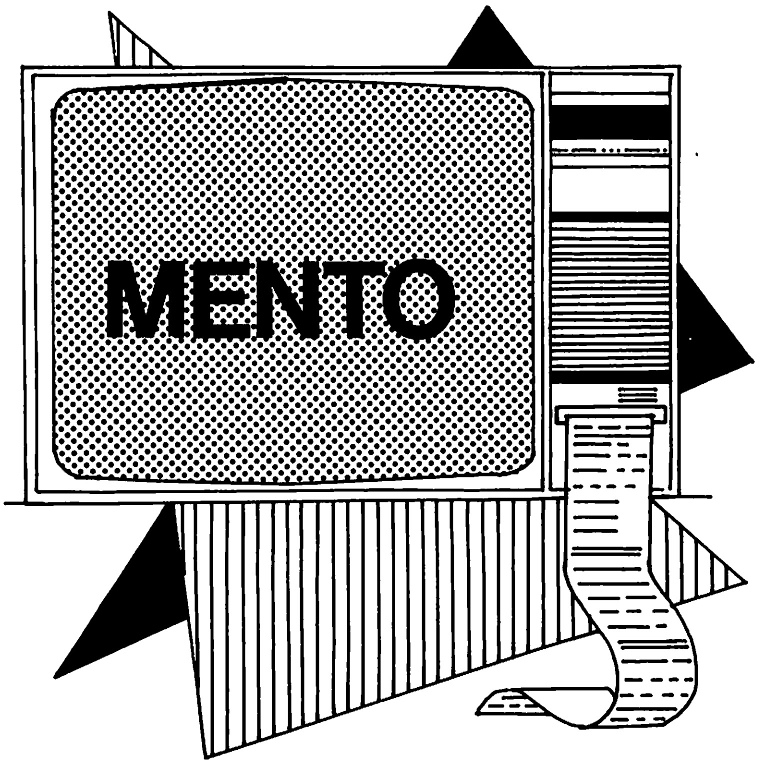
```
110 CALL CHAR(131,"FF00FF00FF00FF00")
120 CALL CHAR(132,"5555555555555555")
130 CALL CHAR(133,"007F405F50575455")
140 CALL CHAR(134,"00FE02FA0AEA2AAA")
150 CALL CHAR(135,"AA2AEA0AFA02FE00")
160 CALL CHAR(136,"555457505F407F00")
170 CALL CHAR(144,"F0BCDFABD73B0F03")
180 CALL CHAR(152,"1818107C1028286C")
190 CALL CHAR(48,"C3C3C3C3C3C3FFFF")
200 CALL CHAR(112,"995A001818004281")
210 CALL CHAR(96,"99DBBD9999A5C3FF")
220 CALL CHAR(40,"40E04040F048241E")
230 FOR Z=1 TO 5
240 CALL CLEAR
250 CALL HCHAR(1,1,133)
260 CALL HCHAR(1,2,129,30)
270 CALL HCHAR(1,32,134)
280 CALL VCHAR(2,32,130,22)
290 CALL HCHAR(24,32,135)
300 CALL HCHAR(24,2,131,30)
310 CALL VCHAR(2,1,132,22)
320 CALL HCHAR(24,1,136)
330 CALL HCHAR(1,16,53)
340 LIFE=5
350 RANDOMIZE
360 FOR I=1 TO 200
370 R=INT(RND*22)
380 C=INT(RND*29)
390 CALL HCHAR(R+2,C+2,144,2)
400 NEXT I
410 A=2
420 B=2
430 CALL COLOR(3,7,16)
440 CALL COLOR(12,11,4)
450 CALL COLOR(16,16,4)
460 CALL COLOR(13,2,3)
470 CALL COLOR(14,2,3)
480 CALL HCHAR(23,31,48)
490 CALL HCHAR(22,31,32)
500 CALL KEY(0,K,S)
510 CALL HCHAR(A,B,152)
520 IF S=0 THEN 500
530 IF K=76 THEN 540 ELSE 600
540 CALL GCHAR(A,B+1,X)
550 IF X=144 THEN 750
560 IF B+1=32 THEN 500
570 B=B+1
580 CALL HCHAR(A,B-1,32)
590 GOTO 500
```

```
600 IF K=77 THEN 610 ELSE 500
610 CALL GCHAR(A+1,B,X)
620 IF X=144 THEN 750
630 IF A+1=24 THEN 500
640 A=A+1
650 CALL HCHAR(A-1,B,32)
660 IF A=23 THEN 670 ELSE 500
670 IF B=31 THEN 680 ELSE 500
680 CALL HCHAR(23,31,96)
690 PRINT "********YOU DID IT********"
700 CALL SOUND(150,784,0)
710 FOR H=1 TO 200
720 NEXT H
730 NEXT Z
740 GOTO 1040
750 CALL HCHAR(A,B,112)
760 CALL COLOR(11,7,4)
770 CALL SOUND(100,-3,0)
780 FOR I=1 TO 15
790 NEXT I
800 CALL SOUND(100,-3,0)
810 FOR I=1 TO 40
820 NEXT I
830 CALL SOUND(100,-3,0)
840 LIFE=LIFE-1
850 IF LIFE<0 THEN 910
860 CALL HCHAR(A-1,B-1,32,3)
870 CALL HCHAR(A,B-1,32,3)
880 CALL HCHAR(A+1,B-1,32,3)
890 CALL HCHAR(1,16,LIFE+48)
900 GOTO 500
910 CALL HCHAR(A,B,40)
920 FOR I=1 TO 3
930 CALL SOUND(400,262,0)
940 NEXT I
950 CALL SOUND(400,311,0)
960 CALL SOUND(300,294,0)
970 FOR I=1 TO 2
980 CALL SOUND(70,247,0)
990 CALL SOUND(300,262,0)
1000 NEXT I
1010 PRINT "TOO MANY TOUCHES !":"YOU ARE DEAD
1020 END
1030 GOTO 1030
1040 CALL CLEAR
1050 PRINT TAB(6);"CONGRATULATIONS!"::
1060 PRINT TAB(6);"YOU HAVE ESCAPED"::::
1070 GOTO 1070
1080 END
```

MENTO

This program tests your powers of mental arithmetic and demonstrates that your computer can apparently read minds.

A series of instructions will appear on the screen. You have to carry out each instruction, then press the ENTER key. At the end, the clever computer will tell you not only how old you are, but how much change you have in your pocket!

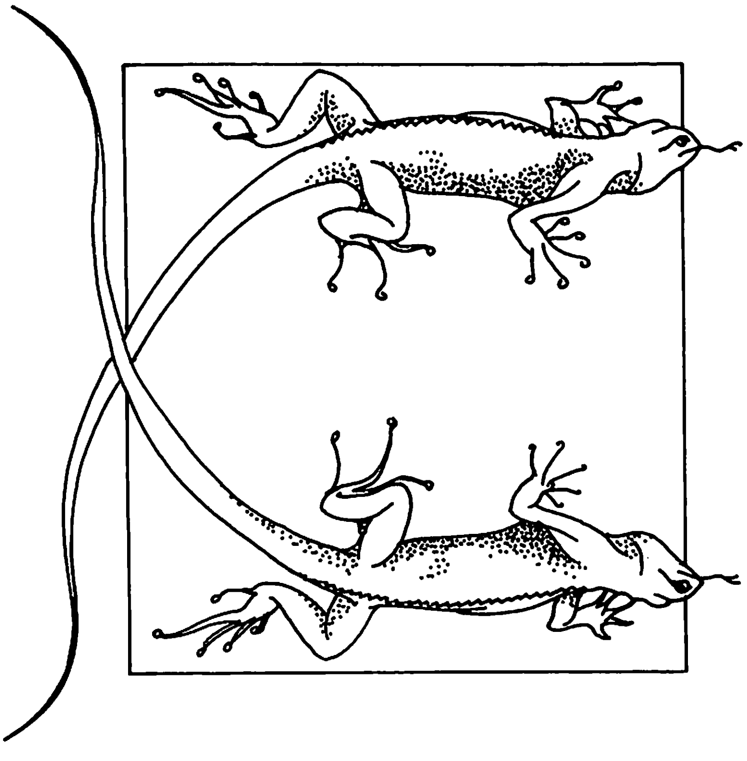


```
10 REM  MENTO
20 GOSUB 320
30 PRINT "MULTIPLY YOUR AGE BY"
40 PRINT "TWO, THEN ADD FIVE"
50 PRINT :::::
60 GOSUB 300
70 PRINT "NOW MULTIPLY THAT"
80 PRINT "BY FIFTY, AND"
90 PRINT "SUBTRACT 365"
100 PRINT :::::
110 GOSUB 300
120 PRINT "NOW ADD THE AMOUNT"
130 PRINT "OF CHANGE IN YOUR"
140 PRINT "POCKET"
150 PRINT ::::::
160 GOSUB 300
170 PRINT "NOW GIVE ME THE"
180 PRINT "NUMBER YOU'VE ENDED"
190 PRINT "UP WITH"
200 PRINT ::::::
210 INPUT A
220 A=A+115
230 B=INT(A/100)
240 A=A-100*B
250 GOSUB 320
260 PRINT "YOU HAVE";A;"CHANGE"
270 PRINT
280 PRINT "AND YOU'RE";B;"YEARS OLD"
290 END
300 PRINT
310 INPUT Z$
320 CALL CLEAR
330 FOR T=1 TO 1000
340 NEXT T
350 RETURN
```

RACE OF THE LIZARD MEN

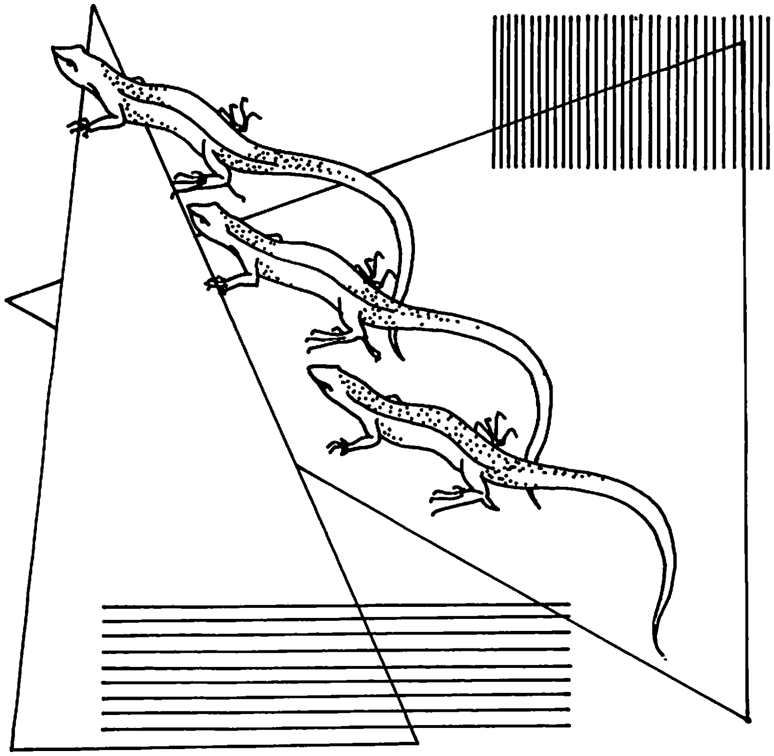
Wilton J. Faberge's user-defined graphics (which do, I suppose, look vaguely like 'lizard men') trundle slowly across the screen in a race for the winner's laurel wreath. Place your bets, and then see if lizard man one or two gets across the screen first.

If you wish to change their appearance, modify the strings A$ to H$ in lines 500 to 570.



```
10 REM  RACE OF THE LIZARD MEN
20 RANDOMIZE
30 CALL CLEAR
40 GOSUB 490
50 A1=6
60 A2=16
70 B1=2
80 C1=2
90 C2=2
100 B2=2
110 J=0
120 IF 2*(INT(J/2))=J THEN 180
130 C=128
140 D=130
150 E=129
160 F=131
170 GOTO 220
180 C=144
190 D=146
200 E=145
210 F=147
220 J=J+1
230 CALL HCHAR(A1,B1,C)
240 CALL HCHAR(A1,C1,32)
250 CALL HCHAR(A1,(B1+1),D)
260 CALL HCHAR((A1+1),C1,32)
270 CALL HCHAR((A1+1),B1,E)
280 CALL HCHAR((A1+1),(B1+1),F)
290 CALL HCHAR(A2,B2,C)
300 CALL HCHAR(A2,C2,32)
310 CALL HCHAR(A2,(B2+1),D)
320 CALL HCHAR((A2+1),C2,32)
330 CALL HCHAR((A2+1),B2,E)
340 CALL HCHAR((A2+1),(B2+1),F)
350 C1=B1
360 C2=B2
370 B1=B1+RND
380 B2=B2+RND
390 IF (B1>30)+(B2>30)<0 THEN 410
400 GOTO 120
410 IF B1<B2 THEN 440
420 PRINT ::"THE WINNER IS NUMBER ONE"
430 GOTO 430
440 PRINT ::"THE WINNER IS NUMBER TWO"
450 GOTO 450
460 J=2
470 GOTO 230
480 END
490 REM  DEFINE CHARACTERS
```

```
500 A$="0000000002060404"
510 B$="0000000001030202"
520 C$="C0E000C4C4C4F8C0"
530 D$="C0C0F88888081000"
540 E$="0000000609090501"
550 F$="0101120600010602"
560 G$="607000808080C0B0"
570 H$="0000804080000000"
580 CALL CHAR(128,A$)
590 CALL CHAR(129,B$)
600 CALL CHAR(130,C$)
610 CALL CHAR(131,D$)
620 CALL CHAR(144,E$)
630 CALL CHAR(145,F$)
640 CALL CHAR(146,G$)
650 CALL CHAR(147,H$)
660 RETURN
```



ENCHANTED FOREST

This 'mini-Adventure', based on a program created by Toni Baker and converted for the TI by Tim Hartnell, places you within a strange forest...made up entirely of numbered triangles.

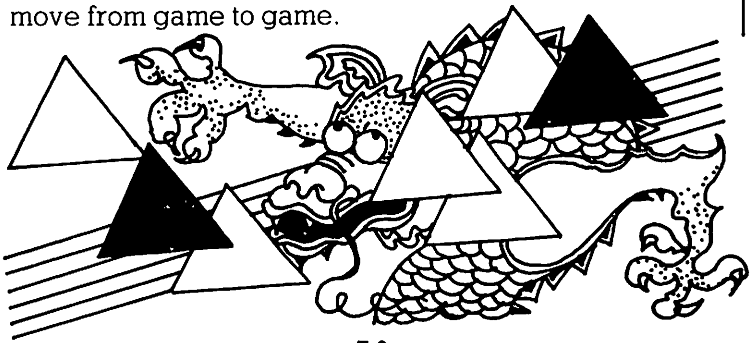
The occupants of the Enchanted Forest, and the effect on you when you move into a triangle where they are, are as follows:

FAIRIES — these will not harm you, but will move you at random to a new triangle within the forest.

GOBLINS — these are definitely nasty, and will kill you if you have the temerity to move into any of their sectors.

THE DRAGON — land in his triangle, and you're a goner!

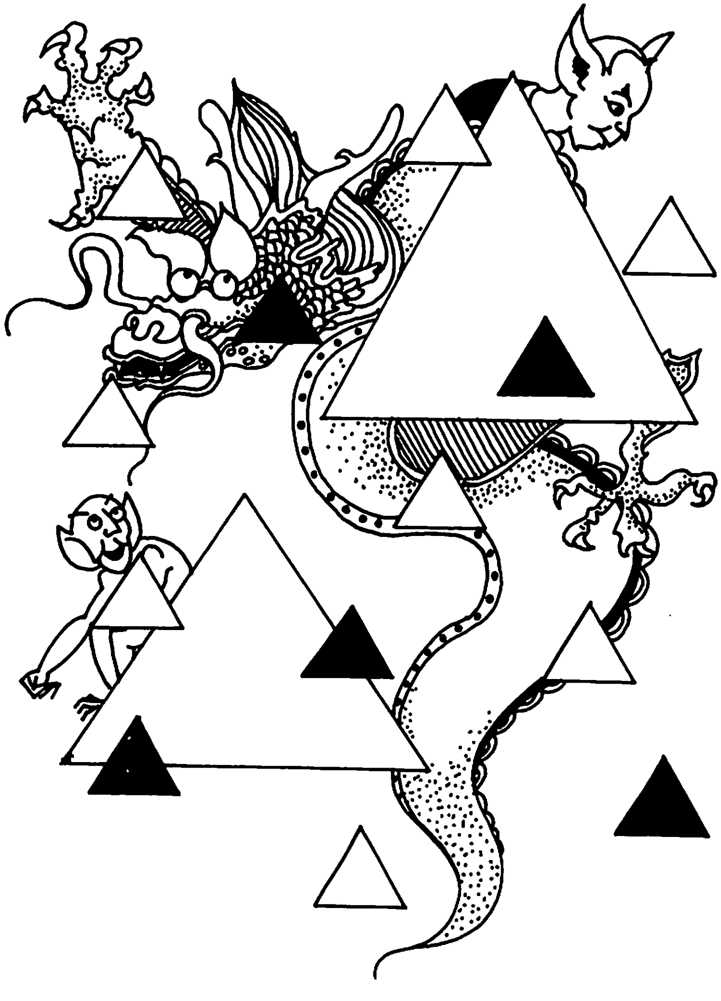
The aim of the game is to kill the dragon. You have a limited number of arrows (set at five in line 40). You fire them into any segment (which you think contains the dragon) by entering the number of the triangle, preceded by a minus sign (that is, you enter -41 to shoot into triangle 41). You'll soon become quite adept at working out where the dragon is likely to be — although he does move from game to game.



```
10 REM  ENCHANTED FOREST
20 DIM A(30),B(30)
30 CALL CLEAR
40 G=5
50 GOSUB 1000
60 PRINT ::
70 FOR Z=0 TO 30
80 A(Z)=0
90 NEXT Z
100 FOR Z=0 TO 10
110 Q=INT(RND*30)+1
120 IF A(Q)=1 THEN 110
130 A(Q)=1
140 B(Z)=Q+35
150 NEXT Z
160 FOR I=1 TO 20
170 PRINT "***************************"
180 NEXT I
190 GOSUB 1000
200 PRINT ::"THERE ARE FAIRIES HERE "
210 FOR Z=1 TO 500
220 NEXT Z
230 X=34+2*(RND*16)+1
240 X=INT(X)
250 Y=7
260 CALL CLEAR
270 FOR I=1 TO 15
280 PRINT
290 NEXT I
300 PRINT :::"YOU ARE NOW IN SECTOR";X
310 FOR J=1 TO 200
320 NEXT J
330 GOSUB 1000
340 Q=-1
350 FOR Z=0 TO 10
360 IF B(Z)<>X THEN 380
370 Q=INT(Z/5)
380 NEXT Z
390 IF Q=0 THEN 200
400 IF Q=1 THEN 850
410 IF Q=2 THEN 880
420 FOR U=1 TO 5
430 PRINT
440 NEXT U
450 PRINT "***************************"
460 PRINT ::
470 PRINT "YOU CAN MOVE TO";X-1;X+1;X+Y
480 GOSUB 1000
490 PRINT ::
```

```
500 FOR Z=0 TO 2
510 A(Z)=0
520 NEXT Z
530 FOR Z=0 TO 10
540 D=B(Z)-X
550 IF ABS(D)=1 THEN 900
560 IF D=Y THEN 900
570 NEXT Z
580 D=ABS(D)
590 IF (D=1)+(D=6)+(D=8)<O THEN 920
600 PRINT ::"##########################"
610 FOR J=1 TO 500
620 NEXT J
630 PRINT ::
640 IF A(0)=1 THEN 940
650 IF A(1)=1 THEN 960
660 GOSUB 1000
670 IF A(2)=1 THEN 980
680 Q=2
690 PRINT :
700 INPUT "YOUR MOVE? ":M
710 FOR J=1 TO 500
720 NEXT J
730 IF M<0 THEN 770
740 X=M
750 Y=-Y
760 GOTO 260
770 IF M=-B(10)THEN 410
780 G=G-1
790 PRINT ::
800 PRINT "YOU HAVE";G;" ARROWS LEFT"
810 PRINT
820 IF G>0 THEN 700
830 PRINT :"SO THE GAME IS OVER"
840 END
850 PRINT :"AND THE GOBLINS HAVE"
860 PRINT "KILLED YOU"
870 END
880 PRINT :"YOU HAVE FOUND THE DRAGON! "
890 END
900 A(INT(Z/5))=1
910 GOTO 570
920 A(2)=1
930 GOTO 600
940 PRINT "FAIRIES NEARBY"
950 GOTO 680
960 PRINT "GOBLINS NEARBY "
970 GOTO 680
980 PRINT ">> DRAGON NEARBY <<"
```

```
990 GOTO 680
1000 FOR MUSIC=1 TO INT(RND*6)+1
1010 NOTE=INT(RND*263)+262
1020 TIME=INT(RND*90)+20
1030 CALL SOUND(TIME,NOTE,0,
    (NOTE+9),0,(NOTE-6),0)
1040 NEXT MUSIC
1050 RETURN
```

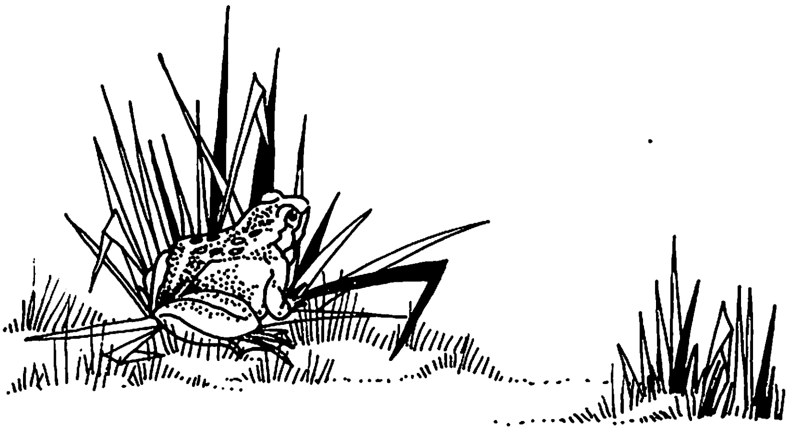


FROG PLAGUE

The frog's life is in your hands in this program from Damon Pillinger.

As the REM statements point out, you use the 'S' and 'D' keys to move left and right, in an attempt to avoid the objects moving up the screen towards you. You have to keep out of the way of leaves, a motorboat, rocks, logs and (hardest of all) bridges.

This is a game which, despite all your efforts, you are bound to lose sometime. Walls on the screen constantly move in on you, making it harder and harder to avoid the dangerous objects.



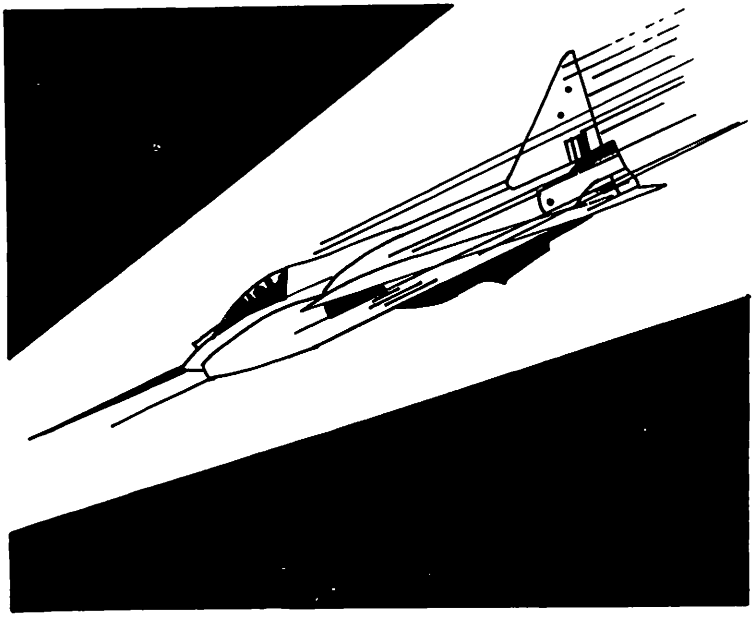
```
10 REM   **** FROG PLAGUE ****
20 REM   **** GAMETRONICS ***
30 REM   *** DAMON PILLINGER ***
40 REM
50 REM  USE S AND D TO MOVE
60 REM  LEFT AND RIGHT. THE OBJECTS
70 REM  YOU ARE AVOIDING ARE:
80 REM    LEAVES, MOTORBOAT
90 REM    ROCKS, LOGS AND BRIDGES
100 REM
110 REM  THE WALLS MOVE IN AS
```

```
120 REM  THE GAME CONTINUES...
130 CALL CLEAR
140 CALL SCREEN(2)
150 SCORE=0
160 CALL COLOR(9,14,2)
170 CALL COLOR(10,7,2)
180 CALL COLOR(11,11,2)
190 CALL COLOR(12,8,2)
200 CALL COLOR(4,2,15)
210 CALL COLOR(2,3,2)
220 LEV=2
230 T=09
240 CALL CHAR(40,"0000C33C187E9918")
250 CALL CHAR(104,"0070FE7E1E1E0101")
260 CALL CHAR(96,"1010383838383810")
270 CALL CHAR(112,"303C3C3C783C3C00")
280 CALL CHAR(57,"FF10101010101 0FF")
290 CALL CHAR(120,"0001501878F8FA20")
300 CALL CHAR(56,"")
310 CALL CLEAR
320 FOR LEV=1 TO 11 STEP .02
330 PRINT TAB(RND*((22-(LEV*2))+10));
    CHR$((INT((RND*4)))*8+96)
340 IF (LEV>0)*(INT(RND*15)=10)THEN 530
350 CALL HCHAR(23,((22-(LEV*2)+10)+1),56)
360 CALL HCHAR(8+LEV,T-1,32,4)
370 CALL HCHAR(9+LEV,T-1,32,4)
380 CALL GCHAR(10+LEV,T,M)
390 SCORE=SCORE+LEV/4
400 IF (M<>32)*(M<>40)THEN 520
410 CALL HCHAR(10+LEV,T,40)
420 CALL KEY(3,X,Y)
430 IF Y=0 THEN 500
440 IF X<>83 THEN 470
450 IF T<2 THEN 470
460 T=T-1
470 IF X<>68 THEN 500
480 IF T>((22-(LEV*2)+10))THEN 500
490 T=T+1
500 NEXT LEV
510 GOTO 310
520 GOTO 550
530 CALL HCHAR(21,1,57,
    (INT(RND*((22-(LEV*2))+10)))+1)
540 GOTO 350
550 CALL CLEAR
560 PRINT "YOU ARE DEAD"
570 PRINT :"YOUR SCORE IS";SCORE
580 PRINT ,,,,,,
```

KAMIKAZE PILOT

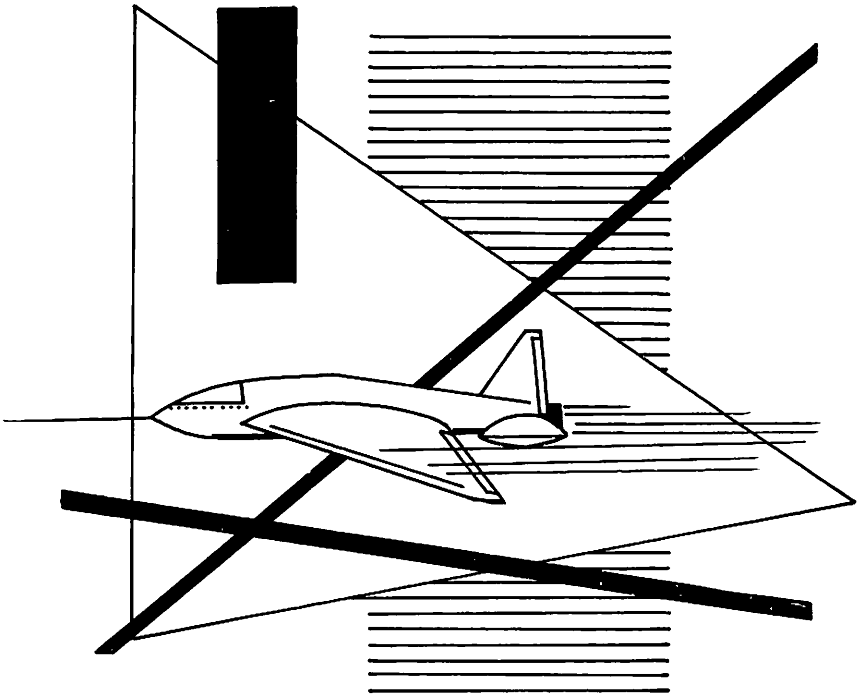
Take your fate in your hands and steer this tiny plane down a twisting, turning tunnel. The plane will drift to one side throughout the game. As you attempt to keep it under control, touch any key and you will correct the drift.

There is a choice of nine levels, with the higher numbers representing the higher (almost impossible) levels. Once you think you have developed some skill at this game, make it harder by reducing the number of spaces (the game is listed with seven spaces) in lines 260 and 270.



```
10 REM  KAMIKAZE PILOT
20 GOTO 50
30 SIDE=SIDE-1
40 GOTO 320
50 CALL CLEAR
60 PRINT "ENTER A NUMBER FROM 1 TO 9"
70 PRINT :"1 IS EASIEST, 9 IS HARDEST"
80 CALL KEY(3,KY,KK)
90 IF KK=0 THEN 80
100 GOTO 360
110 Y=10*Y
120 CALL CLEAR
130 SC=0
140 GOSUB 480
150 SIDE=16
160 DOWN=12
170 CALL GCHAR(17,SIDE,X)
180 IF X=65 THEN 620
190 CALL HCHAR(DOWN,SIDE,66)
200 FOR T=1 TO Y
210 NEXT T
220 D=INT(D+RND*3-RND*3)
230 IF D<3 THEN 540
240 IF D>29 THEN 560
250 CALL HCHAR(DOWN,SIDE,32)
260 PRINT TAB(D);"A       A"
270 PRINT TAB(D);"A       A"
280 SC=SC+27
290 CALL KEY(3,KY,KK)
300 IF KK=0 THEN 30
310 SIDE=SIDE+2
320 IF SIDE<4 THEN 580
330 IF SIDE>28 THEN 600
340 GOTO 170
350 END
360 Y=-(KY=131)
370 Y=-2*(KY=132)
380 Y=-3*(KY=135)
390 Y=-4*(KY=130)
400 Y=-5*(KY=142)
410 Y=-6*(KY=140)
420 Y=-7*(KY=129)
430 Y=-8*(KY=134)
440 Y=-9*(KY=143)
450 CALL CLEAR
460 CALL SCREEN(5)
470 GOTO 110
480 A$="49221C08493E1C08"
490 B$="0810202018040810"
```

```
500 CALL CHAR(66,A$)
510 CALL CHAR(65,B$)
520 D=10
530 RETURN
540 D=3
550 GOTO 260
560 D=29
570 GOTO 260
580 SIDE=4
590 GOTO 320
600 SIDE=28
610 GOTO 340
620 PRINT ::"THE END!!"
630 CALL HCHAR(DOWN,SIDE,66)
640 CALL SOUND(2000,440,2,659,2,880,2)
650 CALL SOUND(2000,-2,2)
660 PRINT ::
670 PRINT "YOU SCORED";SC
680 FOR T=1 TO 1000
690 NEXT T
```

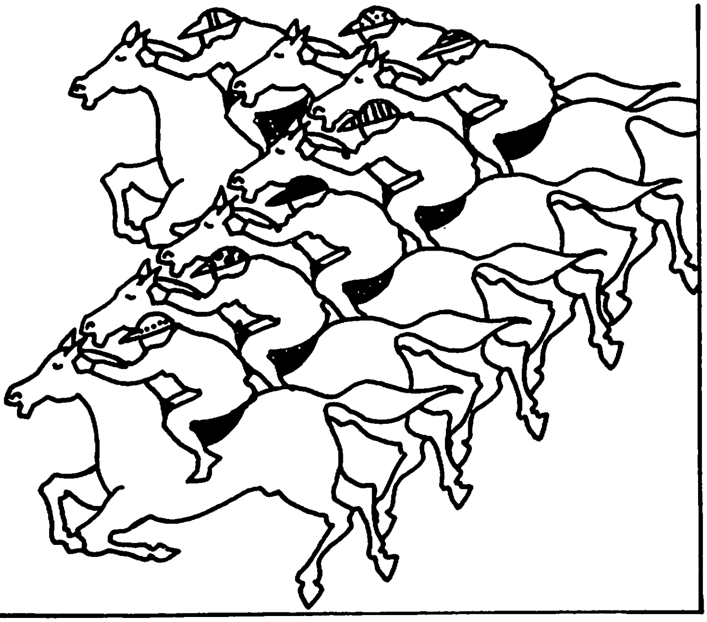


DARING DAMON

You can now indulge your wicked, secret vice of gambling. The program demonstrates how effective moving graphics can be in ordinary BASIC.

The game is a horse-racing one. After you have placed your bets, the race will begin. The barriers move past your horses as they make their way down the track (extremely effective graphics are used for the horses' legs).

Full instructions are contained within the program. If you want to alter the speed at which the horses run, modify the random number generated within the program. You can also change the horses' names by changing the DATA statements.



```
10 REM  *** DARING DAMON ***
20 REM
30 REM  * BY D&D
40 DIM ML(10),BET(10),HOR(10)
50 RANDOMIZE
60 CALL CLEAR
70 CALL SCREEN(3)
80 FOR T=1 TO 10
90 ML(T)=2000
100 NEXT T
110 FOR T=9 TO 12
120 CALL COLOR(T,2,4)
130 NEXT T
140 PRINT "welcome to the texan race":
       "meeting ":,,"your credit with
        daring":"damon is set at
    $2000":"there is no maximum bet."
150 GOSUB 1230
160 CALL CLEAR
170 INPUT "HOW MANY PLAYERS  ":PLA
180 IF PLA>8 THEN 170
190 CALL CLEAR
200 PRINT "hi,":"   i am daring damon.
     i am":"called this because of my
     incredibly high odds.",,,," be
     warned i have inside    tips."
210 PRINT "so heed my odds.but i have
     been known to be wrong."
220 GOSUB 1230
230 DIM HNA$(10),ODD(6)
240 HNA$(1)="NAG HEAP    "
250 HNA$(2)="COSMIC RUN  "
260 HNA$(3)="PRINCE      "
270 HNA$(4)="DASSY       "
280 HNA$(5)="FFFLYER     "
290 HNA$(6)="EXTENDER    "
300 HNA$(8)="VECTOR      "
310 HNA$(9)="FLAT FEET   "
320 HNA$(10)="MULE        "
330 HNA$(7)="DOPPY       "
340 CALL CLEAR
350 CALL SCREEN(13)
360 PRINT "     *DARING DAMON*"
370 PRINT ,,,,"NO.NAME         ODDS"
380 CHANCE=-CHANCE
390 RR$="1111111111"
400 PRINT "**********************",,
410 FOR T=1 TO 5
```

```
420 J=INT(RND*10)+1
430 IF SEG$(RR$,J,1)<>"1" THEN 420
440 RR$=SEG$(RR$,1,J)&"2"&SEG$(RR$,J+1,10)
450 ODD(T)=INT(RND*(30))+20
460 DT$="*"&STR$(T)&"**"&(SEG$(HNA$(J),
    1,20))&"*"&STR$(ODD(T))&"/1"
470 PRINT DT$:
480 NEXT T
490 PRINT ,,
500 FOR T=1 TO PLA
510 PRINT "PLAYER ";T;" WHICH HORSE?"
520 INPUT HOR(T)
530 IF HOR(T)<6 THEN 560
540 PRINT "THERE ARE ONLY 5 HORSES
        IN THE RACE.  STUPID....."
550 GOTO 520
560 NEXT T
570 CALL CLEAR
580 FOR T=1 TO PLA
590 PRINT "PLAYER ";T;"WHAT IS YOUR BET"
600 IF ML(T)=0 THEN 650
610 INPUT BET(T)
620 IF BET(T)<=ML(T)THEN 650
630 PRINT ,,"BE VERY LUCKY YOU
             ONLY HAVE $";ML(T)
640 GOTO 610
650 NEXT T
660 GOSUB 1230
670 CALL CLEAR
680 DIM HOR$(2,2),RL$(2)
690 HOR$(1,1)="0000009F7F1F2040"
700 HOR$(1,2)="0040F0C080008040"
710 HOR$(2,1)="000000030F030201"
720 HOR$(2,2)="00081EF8F0E01020"
730 RL$(1)="0000FFAAFF444444"
740 RL$(2)="0000FF55FF111111"
750 GAT1$="0003010101030408"
760 GAT3$="00C08080800C02010"
770 GAT2$="1121C142FF000000"
780 GAT4$="88848342FF000000"
790 FOR T=1 TO 6
800 A(T)=4
810 NEXT T
820 CALL CHAR(104,RL$(1))
830 CALL HCHAR(7,1,104,32)
840 CALL HCHAR(14,1,104,32)
850 CALL CHAR(112,GAT1$)
860 CALL CHAR(114,GAT2$)
```

```
870 CALL CHAR(113,GAT3$)
880 CALL CHAR(115,GAT4$)
890 CALL HCHAR(7,5,115)
900 CALL CHAR(96,"00")
910 CALL COLOR(9,2,4)
920 CALL COLOR(10,2,3)
930 CALL HCHAR(8,1,96,192)
940 FOR T=1 TO 50000
950 FOR F=1 TO 5
960 CALL HCHAR(8+(F-1),INT(A(F)),97)
970 CALL HCHAR(8+(F-1),INT(A(F))+1,98)
980 CALL HCHAR(8+(F-1),INT(A(F))-3,96,3)
990 F1=RND*3
1000 OD1=(ODD(F)/60)
1010 CHANCE=F1-OD1
1020 IF CHANCE>0 THEN 1040
1030 CHANCE=-(CHANCE)
1040 A(F)=A(F)+CHANCE
1050 IF T>2 THEN 1100
1060 CALL HCHAR(6,4,112)
1070 CALL HCHAR(6,5,113)
1080 CALL HCHAR(7,4,114)
1090 CALL HCHAR(7,5,115)
1100 IF T<>5 THEN 1150
1110 CALL HCHAR(6,4,32)
1120 CALL HCHAR(7,4,104)
1130 CALL HCHAR(6,5,32)
1140 CALL HCHAR(7,5,104)
1150 GH=INT((SIN(F/6))*2)+1
1160 CALL CHAR(98,HOR$(GH,2))
1170 CALL CHAR(97,HOR$(GH,1))
1180 CALL CHAR(104,RL$(GH))
1190 IF A(F)>20 THEN 1270
1200 IF A(F)>29 THEN 1320
1210 NEXT F
1220 NEXT T
1230 PRINT ,,"PRESS ANY KEY TO CONTINUE"
1240 CALL KEY(3,X,Y)
1250 IF Y=0 THEN 1240
1260 RETURN
1270 CALL HCHAR(6,28,112)
1280 CALL HCHAR(6,29,113)
1290 CALL HCHAR(7,28,114)
1300 CALL HCHAR(7,29,115)
1310 GOTO 1200
1320 FOR J=1 TO 3
1330 HI=0
1340 FOR T=1 TO 5
1350 IF A(T)<A(HI)THEN 1370
```

```
1360 HI=T
1370 NEXT T
1380 WI(J)=HI
1390 A(HI)=0
1400 GOTO 1490
1410 NEXT J
1420 FOR T=1 TO PLA
1430 ML(T)=ML(T)-INT(BET(T))
1440 IF ML(T)>=0 THEN 1470
1450 ML(T)=0
1460 PRINT "PLAYER ";T;" ROTTE
     N EGGS.  YOU HAVE BEEN BUSTED."
1470 NEXT T
1480 GOTO 1590
1490 FOR GG=1 TO 3
1500 FOR T=1 TO PLA
1510 IF HOR(T)<>WI(GG)THEN 1540
1520 PRINT :"PLAYER ";T;"COLLECTS"
1530 ML(T)=ML(T)+(INT(BET(T)*(ODD(T)/(GG+1))))
1540 REM
1550 NEXT T
1560 PRINT ,,
1570 GOTO 1420
1580 NEXT GG
1590 PRINT "NO.  MONEY LEFT",,
1600 PRINT
1610 FOR T=1 TO PLA
1620 PRINT T;"     ";ML(T)
1630 NEXT T
1640 GOTO 220
```

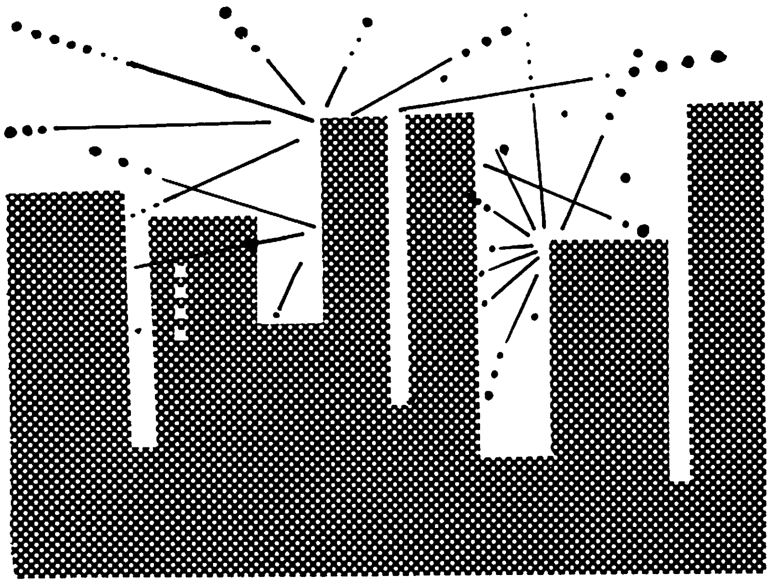
[illegible]

CITY BOMBER

This is a classy adaptation of the old arcade favourite; you drop bombs on a series of high-rise buildings in order to clear a landing strip for your plane.

The program uses some very effective graphics, including a 'random explosion' which changes from game to game. You will find this program a tough nut to crack.

Use any key to fire.



```
10 REM **** CITY BOMBER ****
20 REM
30 REM ***** WRITTEN BY *****
40 REM ** DAMON PILLINGER **
50 REM
60 RANDOMIZE
70 BM$="1A527258DB648F72984D7188305136356
  1376A69AC99E798E7E907BB98516911831757l0
  148700660308306B0E707F0C0C070B707A70707
  E707A70E707DED70D8D0"
```

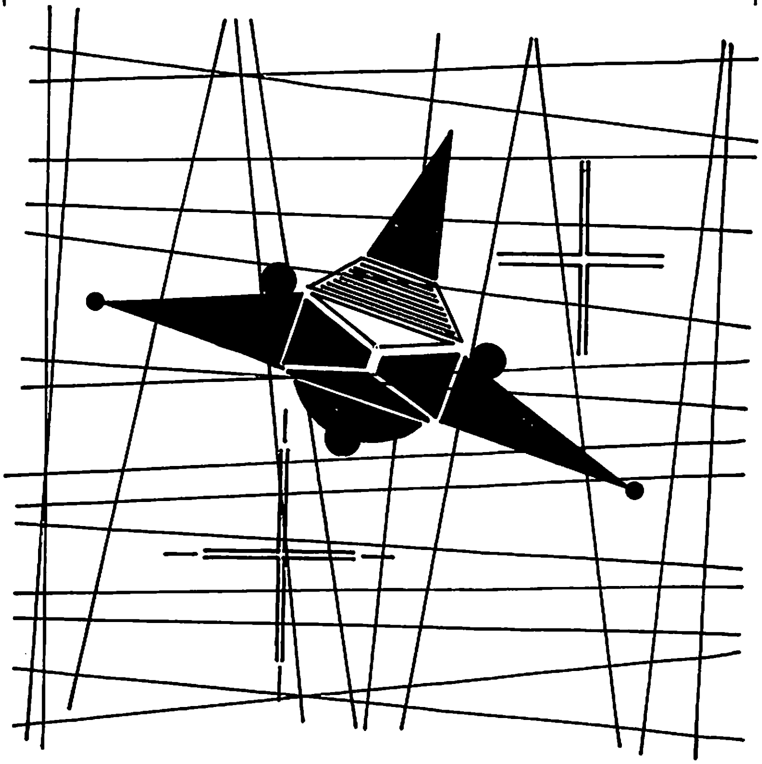
```
80 LEV=1
90 CALL CLEAR
100 FOR T=65 TO 85
110 A$=SEG$(BM$,((RND*100)+1),16)
120 CALL CHAR(T,A$)
130 NEXT T
140 HP=3
150 VP=(LEV)/2+5
160 CALL SCREEN(3)
170 CALL CLEAR
180 FF=32
190 CALL HCHAR(21,1,70,32)
200 CALL COLOR(2,2,14)
210 CALL COLOR(3,2,10)
220 CALL COLOR(4,2,7)
230 CALL COLOR(5,2,9)
240 CALL COLOR(6,2,9)
250 CALL COLOR(7,2,9)
260 CALL COLOR(8,2,9)
270 CALL CHAR(96,"000080C47E7F")
280 CALL CHAR(40,"7F41417F7F414141")
290 CALL CHAR(48,"FF999999FF9999FF")
300 CALL CHAR(56,"18FF3CC3C33CFFFF")
310 CALL CHAR(64,"FF99FF99FF99FF99")
320 FOR T=4 TO 28
330 S=INT(RND*10)
340 CALL VCHAR(20-S,T,
   ((INT(RND*4)*8)+40),S+1)
350 NEXT T
360 CALL KEY(3,X,Y)
370 IF CO<>0 THEN 420
380 IF Y<>0 THEN 400
390 GOTO 520
400 A1=VP+1
410 A2=HP
420 REM
430 CALL HCHAR(A1,A2,32)
440 A1=A1+1
450 IF A1<21 THEN 480
460 CO=0
470 GOTO 360
480 CALL GCHAR(A1,A2,M)
490 CALL HCHAR(A1,A2,111)
500 IF M<>32 THEN 650
510 CO=1
520 CALL HCHAR(VP,HP,96)
530 CALL HCHAR(VP,HP-1,32)
540 CALL SOUND(-100,-6,0)
550 CALL SOUND(-100,-7,1)
```

```
560 IF FF<>32 THEN 740
570 HP=HP+1
580 IF (HP=29)*(VP=20)THEN 800
590 CALL GCHAR(VP,HP,FF)
600 IF HP<=30 THEN 360
610 HP=3
620 VP=VP+1
630 CALL HCHAR(VP-1,28,32,3)
640 GOTO 360
650 FOR T=-9 TO -5
660 CALL HCHAR(A1,A2,T+84)
670 CO=0
680 CALL SOUND(-100,
    (INT(T/3)-3),0,600,5)
690 NEXT T
700 CALL HCHAR(A1,A2,32,1)
710 CALL SOUND(-200,110,0,400,
    0,300,0,-8,3)
720 CALL HCHAR(21,1,70,32)
730 GOTO 360
740 FOR T=1 TO 16 STEP .3
750 CALL SCREEN(T)
760 CALL SOUND(-100,T*10+110,0,-(T/2),1)
770 NEXT T
780 PRINT ,,,,:"YOU CRASHED ON LEVEL ";LEV
790 END
800 FOR Y=1 TO 40
810 CALL SCREEN((RND*5)+1)
820 CALL SOUND(50,110+(Y*10),0,
    -(((RND*7)+1)),1)
830 NEXT Y
840 PRINT ,,,,LEV+1
850 LEV=LEV+2
860 FOR I=1 TO 400
870 NEXT I
880 GOTO 140
```

ARECIBO ATTACK

This is a simple, but highly effective, 'space shoot' game from James Turner that tests your ability to make spatial judgements. You have to try and shoot down a blinking alien as it passes overhead. As the alien gets lower and lower, its descent speed is increased.

To win the game, you have to kill five aliens. The space bar is used to fire; you move left with the 'Z' key, and right with the "." key.



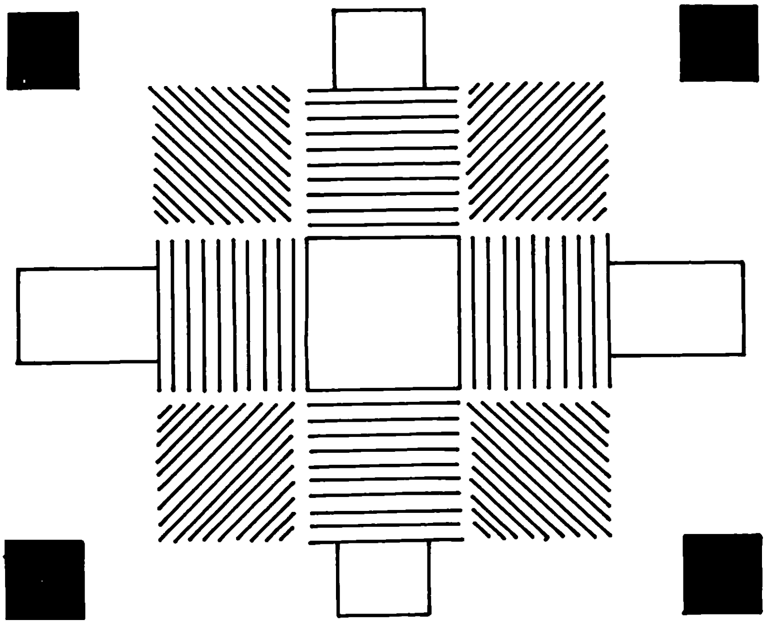
```
10 REM  ARECIBO ATTACK
20 CALL CLEAR
30 PRINT TAB(7);"ARECIBO ATTACK":,,
40 PRINT TAB(13);"By"::
50 PRINT TAB(8);"James Turner"::::::
60 GOSUB 860
70 RANDOMIZE
80 CO=0
90 R=3
100 P=16
110 M=16
120 CALL CLEAR
130 CALL SCREEN(2)
140 S=INT(24*RND)+1
150 ST=INT(32*RND)+1
160 CO=CO+1
170 IF CO=35 THEN 210
180 CALL COLOR(2,16,2)
190 CALL HCHAR(S,ST,46)
200 GOTO 140
210 CALL CHAR(128,"18183C66FFE7BDFF")
220 CALL CHAR(136,"003E49497F491422")
230 CALL COLOR(13,12,2)
240 CALL COLOR(14,9,2)
250 CALL COLOR(12,7,2)
260 CALL KEY(0,K,Q)
270 CALL HCHAR(22,M,128)
280 CALL HCHAR(R,P,136)
290 IF K<>90 THEN 350
300 M=M-2
310 IF M<=1 THEN 320 ELSE 330
320 M=1
330 CALL HCHAR(22,M+2,32)
340 GOTO 410
350 IF K<>46 THEN 400
360 M=M+2
370 IF M>=32 THEN 380 ELSE 390
380 M=32
390 CALL HCHAR(22,M-2,32)
400 IF K=32 THEN 490
410 P=P+1
420 CALL HCHAR(R,P-1,32)
430 IF P<>33 THEN 480
440 P=2
450 GH=MEN+2
460 R=R+GH
470 IF R>=22 THEN 760
480 GOTO 260
490 FOR BP=17 TO R STEP -4
```

```
500 CALL VCHAR(BP,M,124)
510 CALL HCHAR(BP+4,M,32)
520 P=P+1
530 IF P<>33 THEN 560
540 P=2
550 R=R+GH
560 CALL HCHAR(R,P-1,32)
570 CALL HCHAR(R,P,136)
580 CALL SOUND(150,-4,1)
590 CALL GCHAR(BP,M,X)
600 NEXT BP
610 CALL VCHAR(1,M,32,24)
620 CALL VCHAR(1,32,32,24)
630 IF X=136 THEN 640 ELSE 260
640 CALL SOUND(1000,-5,0)
650 CALL SCREEN(10)
660 CALL SOUND(1000,-6,0)
670 CALL SCREEN(2)
680 CALL SOUND(1000,-7,0)
690 CALL SCREEN(10)
700 PRINT "HIT THE MARTIAN!"
710 GOSUB 860
720 MEN=MEN+1
730 IF MEN=5 THEN 820
740 CO=0
750 GOTO 70
760 CALL SCREEN(4)
770 PRINT :"YOU HAVE BEEN INVADED !"
780 PRINT "BETTER LUCK NEXT SHOT !"
790 CALL SOUND(2000,200,0,110,0)
800 GOSUB 860
810 GOTO 70
820 CALL SCREEN(4)
830 PRINT "YOU HAVE TRIUMPHED !"
840 GOSUB 860
850 GOTO 70
860 FOR I=1 TO 2000
870 NEXT I
880 RETURN
```

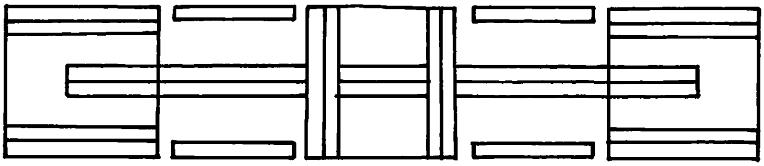
GRAPHIC DEMONSTRA-TIONS

Here are three great graphic demonstration programs, written by Damon Pillinger.

The first, 'Ebauche', draws a fascinating picture pattern on the screen, based on a series of squares. '240 Colours', as its name suggests, shows the full 240 colours which your TI 99/4A can produce. The final program, 'Graphical Girder', uses randomly-produced lines (coming from both the right and the left) to produce an ever-evolving design.

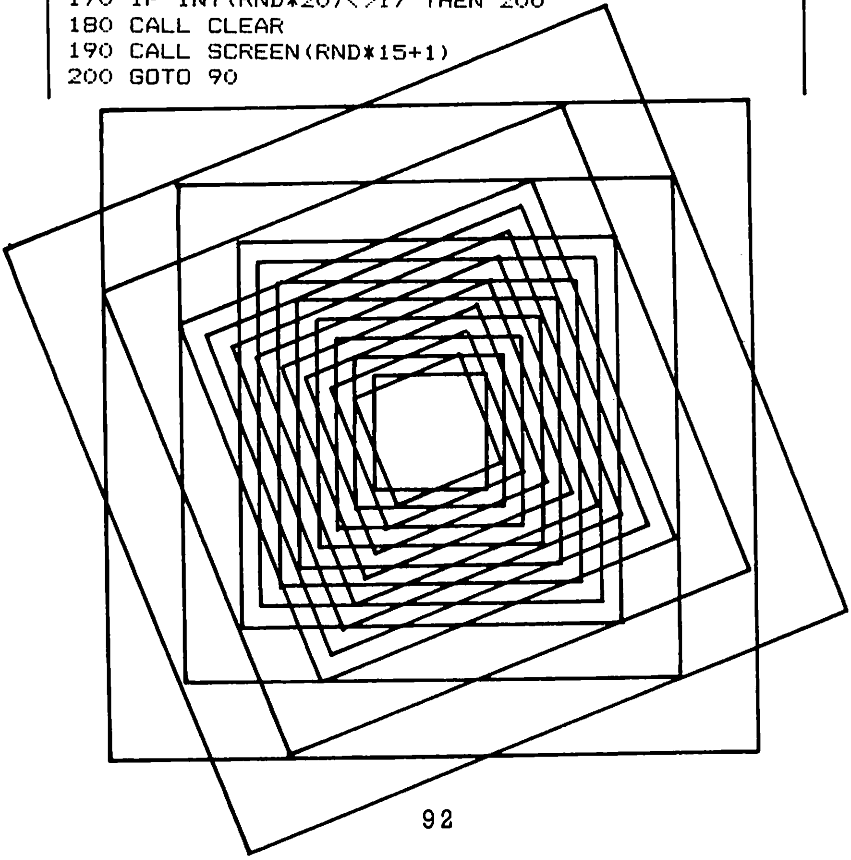


```
10 REM **** EBAUCHE ****
20 CALL SCREEN(4)
30 CALL CLEAR
40 FOR T=2 TO 10
50 CALL CHAR(24+(T*8),"")
60 CALL COLOR(T,T,T)
70 NEXT T
80 FOR T=1 TO 10
90 CALL HCHAR(T,T,32+(T*8),20-(T*2))
100 CALL HCHAR(20-T,T,32+(T*8),20-(T*2))
110 CALL VCHAR(T,T,32+(T*8),20-(T*2))
120 CALL VCHAR(T,19-T,32+(T*8),20-(T*2))
130 NEXT T
140 FOR Y=1 TO 8
150 A(Y)=Y
160 NEXT Y
170 FOR Y=1 TO 8
180 A(Y)=A(Y)+1
190 IF A(Y)<>9 THEN 210
200 A(Y)=1
210 CALL COLOR(Y+1,2,A(Y)+1)
220 NEXT Y
230 GOTO 170
```



```
10 REM **** 240 COLORS ****
20 REM
30 REM *** BY DAMON ***
40 REM
50 CALL CLEAR
60 INPUT "SPEED  (1-300)":SP
70 CALL CHAR(97,"AA55AA55AA55AA55")
80 CALL HCHAR(1,1,97,(32*24))
90 FOR T=2 TO 16
100 FOR J=1 TO 16
110 FOR G=1 TO SP
120 NEXT G
130 CALL COLOR(9,T,J)
140 NEXT J
150 NEXT T
160 CALL CLEAR
170 PRINT ,,,,,,,,"240 COLORS"
```

```
10 REM  GRAPHICAL GIRDER
20 REM
30 CALL SCREEN(2)
40 CALL CLEAR
50 FOR T=2 TO 12
60 CALL CHAR(40+(T-2)*8,"")
70 CALL COLOR(T,T+2,T+2)
80 NEXT T
90 A=3+INT(RND*(28))
100 B=3+INT(RND*(18))
110 CALL VCHAR(1,A,40+INT(((RND*10)*8)),B)
120 CALL HCHAR(B,31-A,
    40+INT(((RND*10)*8)),A)
130 A=INT(RND*28)+3
140 B=INT(RND*18)+3
150 CALL VCHAR(22-B,A,
     40+INT(((RND*10)*8)),B)
160 CALL HCHAR(B,3,40+INT(((RND*10)*8)),A)
170 IF INT(RND*20)<>17 THEN 200
180 CALL CLEAR
190 CALL SCREEN(RND*15+1)
200 GOTO 90
```



CHALLENGER

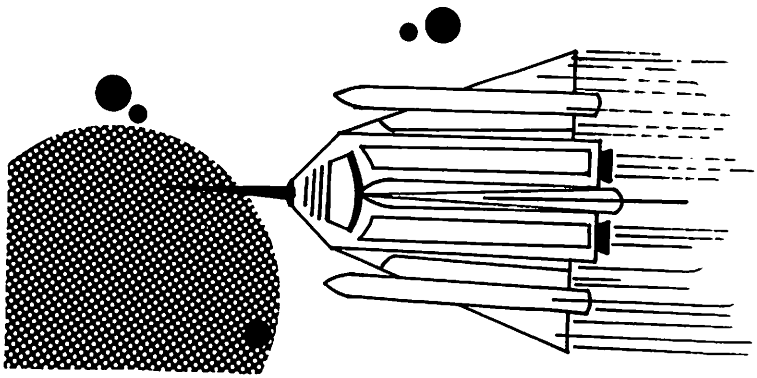
Wayne Southwick wrote this very effective moving graphics game. Written in Extended BASIC, the program makes good use of sprites.

At the start of the game, you will be asked to enter your name. If you have a speech unit fitted, line 210 will inform you: "I think you will like this space-type program." You can delete this line (and 710) if you do not have a speech unit.

After entering your name, you will be given a few instructions. You are the white spaceship near the centre of the screen. Several coloured spacecraft fly past you — at different speeds — and you have to try and dodge them.

If you do nothing, your craft will drift upwards. Touch the 'X' key and you will move downwards. If you hit the red barriers at either the top or the bottom of the screen the game will come to an end.

Your craft is fitted with shields which operate intermittently. If you are really desperate, you can chance flying *through* one of the alien craft. However, if your shields are not operating at that instance, you will be killed — and the game will end.



```
10 REM CHALLENGER
20 REM
30 REM WAYNE SOUTHWICK
40 REM
50 CALL CHARSET :: CALL MAGNIFY(1)
60 CALL CHAR(128,"191A1C1818385898")
70 CALL COLOR(0,16,1)
80 CALL COLOR(13,7,1)
90 CALL COLOR(1,16,1)
100 CALL COLOR(2,16,1)
110 CALL COLOR(3,16,1)
120 CALL COLOR(4,16,1)
130 CALL COLOR(10,16,1)
140 CALL COLOR(5,16,1)
150 CALL COLOR(6,16,1)
160 CALL COLOR(7,16,1)
170 CALL COLOR(8,16,1)
180 CALL COLOR(12,9,1)
190 CALL CLEAR :: CALL SCREEN(2):: DISPLAY
     AT(10,10):"CHALLENGER" :: CALL HCHAR(
     11,12,128,10)
200 PRINT "    PRESS X TO MOVE YOUR
     SPACE-SHIP DOWN"
210 CALL SAY("I+THINK+YOU+WILL+LIKE+
    THIS+SPACE+TYPE+PROGRAM")
220 CALL CHAR(120,"FF7F3F3F7F3F1E04")
230 CALL CHAR(121,"40E1F3F7F3F3F7FF")
240 CALL CLEAR :: CALL SCREEN(2)
250 DISPLAY AT(1,1):"FOR THE FIRST TWO
   100 POINTS" :: DISPLAY AT(3,1):"   YOU WI
   LL GET DIFFERENT" :: DISPLAY AT(5,1)
   :"          ATTACKERS."
260 DISPLAY AT(10,8):"PLEASE ENTER YOUR NAME: "
270 ACCEPT AT(13,8)SIZE(-7)VALIDATE
              (UALPHA)BEEP:A$
280 CALL CLEAR
290 CALL CHAR(136,"FFFFFFFFFFFFFFFF"):
              : CALL COLOR(14,9,1)
300 CALL HCHAR(2,1,136,32)
310 CALL HCHAR(24,1,136,32)
320 CALL HCHAR(3,1,120,32)
330 CALL HCHAR(23,1,121,32)
340 CALL CHAR(43,"DBE7281818 28E7DB")
350 CALL CHAR(111,"0F1E3EFAFA3E1E0F")
360 CALL SPRITE(#1,111,16,90,150)
370 FOR F=2 TO 26 STEP 5
380 CALL SPRITE(#F,43,3,20+F*5,20,0,50)
390 NEXT F
```

```
400 CALL CHAR(45,"00183C7EFF7E7E5A")
410 CALL SPRITE(#7,45,8,160,80,0,30)
420 CALL SPRITE(#8,45,8,120,1,0,20)
430 CALL SPRITE(#9,45,11,50,1,0,20)
440 CALL SPRITE(#10,45,11,65,1,0,20)
450 CALL CHAR(61,"0F1E3EFAFA3E1E0F")
460 LET CHEN=CHEN+1 :: IF CHEN=200 THEN
    800 :: IF CHEN>100 THEN 750
470 CALL COINC(#1,3,150,26,G):
              : IF G=-1 THEN 620
480 CALL COINC(ALL,Q):: IF Q=-1 THEN 620
490 CALL COINC(#1,162,150,10,P):
              : IF P=-1 THEN 620
500 LET TIME=TIME+1 :: DISPLAY AT(1,1):
    A$&", YOUR SCORE IS: "&STR$(TIME)
510 CALL COINC(#1,3,150,26,G):
                   : IF G=-1 THEN 620
520 CALL COINC(ALL,Q):: IF Q=-1 THEN 620
530 CALL COINC(#1,162,150,10,P):
                       : IF P=-1 THEN 620
540 CALL KEY(1,K,S)
550 IF S=0 THEN Y=-15
560 IF K=0 THEN Y=15
570 CALL MOTION(#1,Y,X)
580 CALL COINC(#1,3,150,26,G):
             : IF G=-1 THEN 620
590 CALL COINC(ALL,Q):: IF Q=-1 THEN 620
600 CALL COINC(#1,162,150,10,P):
                       : IF P=-1 THEN 620
610 GOTO 460
620 CALL POSITION(#1,R1,C1)
630 CALL DELSPRITE(ALL):: CALL SOUND
       (680,880,0,-5,0):: CALL SCREEN(2)
640 CALL COLOR(10,11,1)
650 CALL HCHAR(5,2,111,31)
660 CALL VCHAR(5,2,111,18)
670 CALL VCHAR(5,32,111,18)
680 CALL HCHAR(18,2,111,31)
690 DISPLAY AT(12,8):"G A M E  O V E R" :
   : DISPLAY AT(16,8):A$&"'S SCORE IS "&ST
   R$(TIME)
700 DISPLAY AT(1,1):"
              " :: DISPLAY AT(20.4):" WANT TO
    PLAY AGAIN Y OR N.Y"
710 CALL SAY("TO+PLAY+AGAIN+PRESS+ENTER+OR+N
    +THEN+ENTER+FOR+NO")
720 ACCEPT AT(21.28)SIZE(-1)BEEP
    VALIDATE("YN"):G$ :: IF G$="Y" THEN 900
```

```
730 FOR D=1 TO 800 :: NEXT D
740 CALL CLEAR :: END
750 CALL CHAR(43,"183C7EFFFF7E3C18")
760 CALL COINC(ALL,Q):: IF Q=-1 THEN 620
770 CALL CHAR(43,"E7C381000081C3E7")
780 CALL COINC(ALL,Q):: IF Q=-1 THEN 620
790 GOTO 470
800 CALL DELSPRITE(ALL)
810 CALL SPRITE(#1,61,16,90,150)
820 CALL SPRITE(#7,128,8,150,90,0,40)
830 CALL SPRITE(#9,128,4,32,8,0,20)
840 CALL SPRITE(#11,128,6,48,24,0,40)
850 CALL SPRITE(#13,128,8,64,40,0,60)
860 CALL SPRITE(#15,128,10,80,56,0,80)
870 CALL SPRITE(#17,128,12,96,72,0,20)
880 CALL SPRITE(#19,128,14,112,88,0,40)
890 GOTO 470
900 CALL CLEAR :: RUN 240
```

How To Write Better Programs

INVENTING AND DEVELOPING YOUR OWN GAMES PROGRAMS
By Series Editor, Tim Hartnell

It's all very well spending your time typing in programs like those in this book, but there is sure to come a time when you decide you'd like to develop some games programs of your own. In this section of the book, I'd like to discuss a few ideas which may help you write games which you'll both enjoy developing and — more importantly — you and your friends will enjoy playing.

HAVE A CLEAR GOAL IN MIND

Although in many (perhaps most) cases, your computer program will take on a life of its own as you write it, developing away from the concept you had in mind when you started programming, it is important at the outset to have a pretty good idea of what your game will involve.

This is not as obvious a suggestion as you might think. Of course, you'll know if you're developing a 'chase the ghosts around the maze as you eat the power pills' program that you are going to need a different sort of program layout to one which places you inside a Haunted Oak, peopled with gremlins and halflings. But you have to go beyond the basic "I'm going to write me an Adventure" stage to work out such things as (a) what the object of the game will be; (b) what the screen display will look like; (c) what variables, and variable names, you'll need;

(d) the nature of the player input; (e) how 'winning' or 'losing' will be determined; and so on.
Let's look at these one by one.

THE OBJECT OF THE GAME

This can usually be stated very succinctly: "To find the lost treasure of the Aztecs"; "To destroy as many asteroids as possible before running out of ships"; or "To play a game of chess". But even though this stage of the game production can be accomplished very quickly, it should not be overlooked. Get this statement — which might be just a sentence, or may run to a paragraph length or more, if there is more than one 'screen' to be worked through, with a different scenario for each screen — down in writing.

You may well discard the original aim as the program develops, and it looks like the direction it is taking is better than the one you first thought of. Despite this, it is important to have something concrete to aim at, to stop you wasting hour after hour doodling aimlessly.

THE SCREEN DISPLAY

I've found that making a sketch, or sketches, of what the display will look like once the program is up and running, is of tremendous benefit. Once you have your drawing, and it doesn't matter how rough it is so long as it shows all the important things you want on the screen, and their relative positions and size, you'll discover the program concept is likely to crystalize.

As well as seeing immediately how you will write parts of the code to achieve the game's aim, you'll get an idea of whether or not the game is even worth writing in the form you had considered. Perhaps the game will be too complex if you leave everything on the screen you were intending to; or maybe most of the screen will be wasted space, so that a more complicated game scenario should be devised.

I've discovered that sketching the proposed screen display before starting to program is particularly useful, especially when creating arcade and simulation games. You get an indication of the variables you'll need, the user-defined graphics, the kind of player inputs which will be most conducive to good player interaction, and so on.

Simulation games, as you probably know, are those in which the computer models an external reality — such as running a cake shop, a war, or an airport — and allows you to experience (after a fashion) what it would be like to take part in such an activity in real life. Simulation games are not particularly difficult to write — in terms of needing clever coding — but instead demand a methodical, painstaking approach to the program.

In my book *The ZX Spectrum Explored* (Sinclair Browne, 1982), there is a program with the unlikely name of 'Workin' for the Man', in which you are running a factory, staffed with a highly-erratic workforce, involved in the manufacture of some mythical product called 'The Zibby'. The player gets a factory report two or three times a week, and from this report has to decide how many staff he or she will hire or (attempt to) fire, how many Zibbies will be the production target for the week, and so on.

This report is the key to the program, and when I wrote the game, I started by making a sketch of how the screen would look. It was a bit like this:

```
FACTORY REPORT: WEEK 5

Capital in hand is $2,657.92


Your stores hold 12 Zibbies worth $169.68
They sell for $14.14 each and cost $7.41
each to make


Workforce is 7 people
Their wages are $41 each and the wage bill
this week is $287
```

Each person can make 10 Zibbies a week,
a total output of 70

Once I had this sketch drawn up, I was ready to go. As you can see, it gives a very good indication of the variables which will be needed. For a start, I know I'll have to control the number of the week, the capital, the contents of the stores (and their value) and so on.

I found that once I'd completed the screen display sketch, the rest of the program was relatively easy to write. Doing a sketch in this way gives you an instant guide to the main variables you'll need.

USE HELPFUL VARIABLE NAMES

I also tend to use variable names which relate in some way to that which they are representing, as it saves having to keep a list of the variables which have been assigned, and what they've been assigned to. For example, I could use WK for week, CH for capital in hand, MZ for the cost of making each Zibby and SZ for the selling price. If Z was the number of Zibbies, I would know that the total value of Zibbies I had was Z (the number of them) multiplied by SZ (their selling price) and it cost me Z multiplied by MZ (their price of manufacture) to make them. My profit, if I sold them all, would then be Z*SZ minus Z*MZ.

If you follow a similar idea, you'll find it is much easier to keep track of what is happening in your program than might otherwise be the case.

THE NATURE OF THE PLAYER INPUT

It's important to make games *easy* and fun to play. It's not good having the best Asteroids-derivative program in the world if players have trouble hitting the fire button because you've placed it right next door to the 'rotate' control.

Many programs which provide 'up', 'down', 'right' and 'left' controls, automatically use arrow or cursor keys, even though these might be most inconvenient for the player to use. Have a look at your keyboard, and see if you can find better ones. I often use "Z" and "M" for programs which need just left and right movement, with the space bar for fire. These keys seem logical to me, and no player time is wasted in learning them, or trying to remember them when the game is underway. In a similar way, I tend to use "A" (for up) and "Z" (for down) for the left hand, and the "greater than" and "less than" keys for left and right (pointing out to the player that the < and > symbols point in the relevant directions).

Use INKEY$ or GET$ whenever you can, to prevent the player from having to use the RETURN or ENTER keys to get the program underway.

HOW THE GAME WILL END

The way the game will be won and lost needs to be defined, and clear to the player. Do you need to blast all the aliens to win, and will you lose automatically if one alien lands, and you've still got ships left, or only if you have no ships left. In a two-player game, is the loser the first player to lose three lives, or seven pieces, or does the game only end when the *difference* between the two scores is three or seven or whatever.

Work this out, and make it very clear to the player. Whether the goal of the game is to clear the left-hand side of the screen of the Screaming Widgies, or to clock up a fortune of $7.3 billion, it must be both clear to the player, and *possible to achieve*. A 'win condition' which can never be achieved on the higher levels of play is most unsatisfactory. No matter how difficult it is to do, you are only defrauding players if you set goals whose achievement is not possible within the constrictions you've put into the game.

I hope these five points may give you a few ideas on how you can go ahead and write programs which will be relatively easy to write, and which will be satisfying for you and your friends to play.

GLOSSARY

A

Accumulator — the place within the computer in which arithmetic computations are performed and where the results of these computations are stored.

Algorithm — the series of steps the computer follows to solve a particular problem.

Alphanumeric — this term is usually used in relation to a keyboard, as in 'it is an alphanumeric keyboard', which means that the keyboard has letters as well as numbers. It is also used to refer to the 'character set' of the computer. The character set comprises the numbers and letters the computer can print on the screen.

ALU (Arithmetic/Logic Unit) — the part of the computer which does arithmetic (such as addition, subtraction) and where decisions are made.

AND — a Boolean logic operation that the computer uses in its decision-making process. It is based on Boolean algebra, a system developed by mathematician George Boole (1815-64). In Boolean algebra the variables of an expression represent a logical operation such as OR and NOR.

ASCII — stands for American Standard Code for Information Exchange, the most widely used encoding system for English language alphanumerics. There are 128 upper and lower case letters, digits and some special characters. ASCII converts the symbols and control instructions into seven-bit binary combinations.

Assembler — a program which converts other programs written in assembly language into machine code (which the computer can understand directly).

Assembly language is a low level programming language which uses easily memorised combinations of two or three letters to represent a particular instruction which the assembler then converts so the machine can understand it. Examples of these are ADD (add), and SUB (subtract). A computer programmed in assembly language tends to work more quickly than one programmed in a higher level language such as BASIC.

B

BASIC — an acronym for Beginners All-Purpose Symbolic Instruction Code. It is the most widely used computer language in the microcomputer field. Although it has been criticised by many people, it has the virtue of being very easy to learn. A great number of BASIC statements resemble ordinary English.

Baud — named after Baudot, a pioneer of telegraphic communications. Baud measures the rate of transfer of information and is approximately equal to one bit per second.

BCD — an abbreviation for Binary Coded Decimal.

Benchmark — a test against which certain functions of the computer can be measured. There are a number of so-called 'standard Benchmark tests', but generally these only test speed. This is rarely the aspect of a microcomputer that is most of interest to the potential buyer.

Binary — a numbering system that uses only zeros and ones.

Bit — an abbreviation for Binary Digit. This is the smallest unit of information a computer circuit can recognise.

Boolean Algebra — the system of algebra developed by mathematician George Boole which uses algebraic notation to express logical relationships (see AND).

Bootstrap — a short program or routine which is read into the computer when it is first turned on. It orients the computer to accept the longer, following program.

Bug — an error in a computer program which stops the program from running properly. Although it is generally used to mean only a fault or an error in a program, the term bug can also be used for a fault in the computer hardware.

Bus — a number of conductors used for transmitting signals such as data instructions, or power in and out of a computer.

Byte — a group of binary digits which make up a computer word. Eight is the most usual number of bits in a byte.

C

CAI — Computer Assisted Instruction.

CAL — Computer Assisted Learning. The term is generally used to describe programs which involve the learner with the learning process.

Chip — the general term for the entire circuit which is etched onto a small piece of silicon. The chip is, of course, at the heart of the microcomputer.

Clock — the timing device within the computer that synchronises its operations.

COBOL — a high level language derived from the words Common Business Orientated Language. COBOL is designed primarily for filing and record-keeping.

Comparator — a device which compares two things and produces a signal related to the difference between the two.

Compiler — a computer program that converts high level programming language into binary machine code so the computer can handle it.

Complement — a number which is derived from another according to specified rules.

Computer — a device with three main abilities or functions:
1) to accept data
2) to solve problems
3) to supply results

CPU — stands for Central Processing Unit. This is the heart of the computer's intelligence, where data is handled and instructions are carried out.

Cursor — a character which appears on the TV screen when the computer is operating. It shows where the next character will be printed. On a computer there are usually 'cursor control keys' to allow the user to move the cursor around the screen.

D

Data — information in a form which the computer can process.

Debug — the general term for going through a program and correcting any errors in it, that is, chasing down and removing bugs (see Bug).

Digital Computer —a computer which operates on information which is in a discrete form.

Disk/Disc — this is a magnetically sensitised plastic disk, a little smaller than a single play record. This is used for storing programs and for obtaining data. Disks are considerably faster to load than a cassette of the same length program. The disk can be searched very quickly while a program is running for additional data.

Display — the visual output of the computer, generally on a TV or monitor screen.

Dot Matrix Printer — a printer which prints either the listing of a program or that which is displayed on the TV screen. Each letter and character is made up of a number of dots. The higher the number of dots per character the finer the resolution of the printer.

Dynamic Memory — a memory unit within the computer which 'forgets' its contents when the power is turned off.

E

Editor — this term is generally used for the routine within the computer which allows you to change lines of a program while you are writing it.

EPROM — stands for Erasable Programmable Read-Only Memory. This is like the ROM in the computer, except that it is fairly easy to load material into an EPROM and it doesn't disappear when you turn the power off. EPROMs must be placed in a strong ultra violet light to erase them.

Error Messages — the information given by a computer where there is a fault in the coding during a part of a program, usually shown by the computer stopping, and printing a word, or a word and numbers, or a combination of numbers only, at the bottom of the screen. This tells you what mistake has been made. Common mistakes include using the letter O instead of zero in a line, or leaving out a pair of brackets, or one of the brackets, in an expression, or failing to define a variable.

F

File — a collection of related items of information organised in a systematic way.

Floppy Disk — a relatively cheap form of magnetic disk used for storing computer information, and so named because it is quite flexible (see Disk/Disc).

Flow Chart — a diagram drawn up before writing a program, in which the main operations are enclosed within rectangles or other shapes and connected by lines, with arrows to represent loops, and decisions written at the branches. It makes writing a program much easier because traps such as infinite loops, or non-defined variables can be caught at an early stage. It may not be worth writing a flow chart for very short programs, but generally a flow chart aids in creating programs.

Firmware — there are three kinds of 'ware' in computers: software 'temporary' programs; hardware like the ROM which contains permanent information; and firmware in which the information is relatively permanent, as in an EPROM (see EPROM).

Flip-Flop — a circuit which maintains one electrical condition until changed to the opposite condition by an input signal.

FORTRAN — an acronym for FORmula TRANslation, this is a high level, problem orientated computer language for scientific and mathematical use.

G

Gate — an electrical circuit which, although it may accept one or more incoming signals, only sends out a single signal.

Graphics — pictorial information as opposed to letters and numbers.

H

Hard Copy — computer output which is in permanent form.

Hardware — the physical parts of the computer (also see software and firmware).

Hexadecimal (Hex) — a numbering system to the base sixteen. The digits zero to nine are used, as well as the letters A, B, C, D, E and F to represent numbers. A equals 10, B equals.11, C equals 12, and so on. Hex is often used by microprocessor users.

Hex Pad — a keyboard designed specifically for entering hexadecimal notation.

High Level Language — a programming language which allows the user to talk to the computer more or less in English. In general, the higher the level of the language (that is, the closer it is to English), the longer it takes for the computer to translate it into a language it can use. Lower level languages are far more difficult for human operators but are generally executed far more quickly.

I

Input — the information fed into the computer via a keyboard, a microphone, a cassette or a disk.

Input/Output (I/O Device) — a device which accepts information or instructions from the outside world, relays it to the computer, and then, after processing, sends the information out in a form suitable for storing, or in a form which could be understood by a human being.

Instruction — data which directs a single step in the processing of information by the computer (also known as a command).

Integrated Circuit — a complete electronic circuit imprinted on a semiconductor surface.

Interface — the boundary between the computer and a peripheral such as a printer.

Interpreter — a program which translates the high level language fed in by the human operator, into a language which the machine can understand.

Inverter — a logic gate that changes the signal being fed in, to the opposite one.

Interactive Routine — part of a program which is repeated over and over again until a specified condition is reached.

J

Jump Instruction — an instruction which tells the computer to go to another part of the program, when the destination of this move depends on the result of a calculation just performed.

K

K — this relates to the size of the memory. Memory is usually measured in 4K blocks. 1K contains 1,024 bytes.

Keyword — the trigger word in a line of programming, usually the first word after the line number. Keywords include STOP, PRINT and GOTO.

L

Language — computer languages are divided into three sections: high level languages, such as BASIC, which are reasonably close to English and fairly easy for humans to use; low level languages, such as Assembler, that use short phrases which have some connection with English (ADD for add and RET for return, for instance); and machine code which communicates more or less directly with the machine.

LCD — this stands for Liquid Crystal Diode. Some computers such as the TRS-80 Pocket Computer use an LCD display.

LED — this stands for Light Emitting Diode. The bright red numbers which are often used on watch or clock displays are made up of LEDs.

Logic — the mathematical form of a study of relationships between events.

Loop — a sequence of instructions within a program which is performed over and over again until a particular condition is satisfied.

M

Machine Language or Machine Code — an operation code which can be understood and acted upon directly by the computer.

Magnetic Disk — see Disk and Floppy Disk.

Mainframe — computers are generally divided into three groups, and the group a computer falls into depends more or less on its size. The computer you are thinking of buying is a microcomputer; medium sized computers are known as minicomputers; and the giant computers that you sometimes see in science fiction movies are mainframe computers. Until 15 years ago mainframe computers were, in practical terms, the only ones available.

Memory — there are two types of memory within a computer. The first is called ROM (read-only memory); this is the memory that comes already programmed on the computer, which tells the computer how to make decisions and how to carry out arithmetic operations. This memory is unaffected when you turn the computer off. The second type is RAM (random access memory). This memory holds the program you type in at the keyboard or send in via a cassette or disk. In most computers the computer 'forgets' what is in RAM when you turn the power off.

Microprocessor — the heart of any computer. It requires peripheral unit interfaces, such as a power supply and input and output devices, to act as a microcomputer.

MODEM — stands for Modulator Demodulator. This is a device which allows two computers to talk to each other over the telephone. The computers usually use a cradle in which a telephone receiver is placed.

Monitor — this has two meanings in computer terms. One meaning is a television-like display. A monitor has no facility for tuning television programs, and usually the picture produced on a monitor is superior to that produced by an ordinary television. The second meaning of a monitor relates to ROM. The monitor of a computer is described as the information it has built in when you buy it. This information allows it to make decisions and carry out arithmetic computations.

Motherboard — a framework to which extra circuits can be added. These extra circuits often give the computer facilities which are not built-in, such as that of producing sound or of controlling a light pen.

MPU — an abbreviation for Microprocessor Unit.

N

Nano-second — a nano-second is one thousand billionth of a second, the unit of speed in which a computer or a memory chip is often rated.

Non-Volatile Memory — memory which is not lost when the computer is turned off. Some of the smaller computers such as the TRS-80 Pocket Computer have non-volatile memory. The batteries hold the program you enter for several hundred hours.

Not — a Boolean logic operation that changes a binary digit into its opposite.

Null String — a string which contains no characters. It is shown in the program as two double quote marks, without anything between them.

Numeric — pertaining to numbers as opposed to letters (that is, alphabetic). Many keyboards are described as being alphanumeric which means both numbers and letters are provided.

O

Octal — a numbering system which uses eight as the base, and the digits 0, 1, 2, 3, 4, 5, 6 and 7. The Octal system is not used very much nowadays in microcomputer fields. The Hexadecimal system is more common (see Hexadecimal).

Operating System — the software or firmware generally provided with the machine that allows you to run other programs.

OR — an arithmetic operation that returns a 1, if one or more inputs are 1.

Oracle — a method of sending text messages with a broadcast television signal. A teletext set is required to decode the messages. Oracle is run by Independent Television Service in the UK, and a similar service — Ceefax — is provided by the BBC.

Output — information or data fed out by the computer to such devices as a TV-like screen, a printer or a cassette tape. The output usually consists of the information which the computer has produced as a result of running a program.

Overflow — a number too large or too small for the computer to handle.

P

Pad — see Keypad.

Page — often used to refer to the amount of information needed to fill one TV screen, so you can talk about seeing a page of a program, the amount of the listing that will appear on the screen at one time.

PASCAL — a high level language.

Peripheral — anything which is hooked onto a computer, for control by the computer, such as a disk unit, a printer or a voice synthesiser.

Port — a socket through which information can be fed out of or in to a computer.

Prestel — the British telecom name for a system of calling up pages of information from a central computer via the telephone and displaying them on a television screen. A similar commercial version in the United States is known as The Source.

Program — in computer terms program has two meanings. One is the list of instructions that you feed into a computer, and the second is used as a verb, as in 'to program a computer'.

PROM — stands for Programmable Read Only Memory. This is a device which can be programmed, and once it is then the program is permanent (also see EPROM and ROM).

R

Random Access Memory (RAM) — the memory within a computer which can be changed at will by the person using the computer. The contents of RAM are usually lost when a computer is turned off. RAM is the memory device that stores the program that you type in and also stores the results of calculations in progress.

Read-Only Memory (ROM) — in contrast to RAM, information in ROM cannot be changed by the user of the computer, and the information is not lost when the computer is turned off. The data in ROM is put there by the manufacturers and tells the computer how to make decisions and how to carry out arithmetic computations. The size of ROM and RAM is given in the unit K (see K).

Recursion — the continuous repetition of a part of the program.

Register — a specific place in the memory where one or more computer words are stored during operations.

Reserved Word — a word that you cannot use for a variable in a program because the computer will read it as something else. An example is the word TO. Because TO has a specific computer meaning, most computers will reject it as a name for a variable. The same goes for words like FOR, GOTO and STOP.

Routine — this word can be used as a synonym for program, or can refer to a specific section within a program (also see Subroutine).

S

Second Generation — this has two meanings. The first applies to computers using transistors, as opposed to first generation computers which used valves. Second generation can also mean the second copy of a particular program; subsequent generations are degraded by more and more noise.

Semiconductor — a material that is usually an electrical insulator but under specific conditions can become a conductor.

Serial — information which is stored or sent in a sequence, one bit at a time.

Signal — an electrical pulse which is a conveyor of data.

Silicon Valley — the popular name given to an area in California where many semiconductor manufacturers are located.

SNOBOL — a high level language.

Software — the program which is entered into the computer by a user which tells the computer what to do.

Software Compatible — this refers to two different computers which can accept programs written for the other.

Static Memory — a non-volatile memory device which retains information so long as the power is turned on, but does not require additional boosts of power to keep the memory in place.

Subroutine — part of a program which is often accessed many times during the execution of the main program. A subroutine ends with an instruction to go back to the line after the one which sent it to the subroutine.

T

Teletext — information transmitted in the top section of a broadcast television picture. It requires a special set to decode it to fill the screen with text information. The BBC service is known as Ceefax, the ITV service as Oracle. Teletext messages can also be transmitted by cable, for example the Prestel service in Britain or The Source in the United States.

Teletype — a device like a typewriter which can send information and also receive and print it.

Terminal — a unit independent of the central processing unit. It generally consists of a keyboard and a cathode ray display.

Time Sharing — a process by which a number of users may have access to a large computer which switches rapidly from one user to another in sequence, so each user is under the impression that he or she is the sole user of the computer at that time.

Truth Table — a mathematical table which lists all the possible results of a Boolean logic operation, showing the results you get from various combinations of inputs.

U

UHF — Ultra High Frequency (300-3000 megaHertz).

Ultra Violet Erasing — Ultra violet light must be used to erase EPROMs (see EPROM).

V

Variable — a letter or combination of letters and symbols which the computer can assign to a value or a word during the run of a program.

VDU — an abbreviation for Visual Display Unit.

Volatile — refers to memory which 'forgets' its contents when the power is turned off.

W

Word — a group of characters, or a series of binary digits, which represent a unit of information and occupy a single storage location. The computer processes a word as a single instruction.

Word-Processor — a highly intelligent typewriter which allows the typist to manipulate text, to move it around, to justify margins and to shift whole paragraphs if necessary on a screen before outputting the information onto a printer. Word-processors usually have memories, so that standard letters and the text of letters, written earlier, can be stored.

BIBLIOGRAPHY

Compiled by Tim Hartnell

Usborne have released a number of very attractive books in their Usborne Computer Books series. Drawing on their vast experience in the field of producing low-priced, highly-coloured, attractive books for young readers, they've produced some books which will enlighten both young and not-so-young readers.

I'll look at three of their titles, three which cover just about the whole field of computer interests:

Information Revolution
(Lynn Myring and Ian Graham, Rigby).
Presenting an eminently readable introduction to the 'revolution' which covers such fields as computers (of course), text information services via the television screen, word processing, 'future phones' and satellite communications, *Information Revolution* is an ideal guide for the person who wants an easy-to-read introduction to the field.

Computer Jargon
(Corinne Stockley and Lisa Watts).
The tone of this book is set by the frontispiece, which has a number of odd little coloured robots sitting around a table laden with computer junk, pointing at each piece saying "This is a disk drive", "This is a digital tracer" (!) and "This is a printer".

Robotics — What Robots Can Do and How They Work
(Tony Potter and Ivor Guild).
This is definitely a candidate for the award of 'the longest

title of the year'. But it is very accurate. Don't be put off by the pretty pictures, as you'll soon discover this book has a lot of solid information. Topics covered include "What robots can and cannot do", "How arm robots work", "How to teach a robot" and "Build your own micro-robot"; this last section actually includes nine pages of circuit diagrams and all to build a little two-motor robot which, following a program typed into your micro, will run about the floor. Robotics is a field of the near future (with personal robots certain to be a bigger craze — when 'real robots' finally arrive — than computers will ever be).

Practise Your BASIC
(Gaby Waters and Nick Cutler).
You'll find this book — which predictably contains a number of exercises, puzzles and problems to solve by writing programs — should be useful in giving you a number of 'core problems' which will run on your computer and which can then be modified to take advantage of your system's special features. Program listings include 'Pattern Puzzles', 'Jumping Man', 'Horse Race', 'Word Editor' and 'Treasure Hunt', a mini-Adventure.

Help With Computer Literacy
(June St Clair Atkinson, Houghton Mifflin).
This is a large format book with an attractive cover, fairly priced for its 122 pages. It appears to be aimed at the early to middle years of secondary education, but contains a lot of material which those teaching younger children could easily adapt. Although it avoids the 'Gee Whiz' approach of the Usborne texts, it uses cartoons and diagrams to get its message across in an inviting manner.

The Interface Computer Encyclopedia
(Ken Ozanne, Interface Publications).
Compiled by a lecturer in mathematics at the NSW Institute of Technology, this work could perhaps be more accurately called 'The Computer Book of Lists', rather

than an encyclopedia. It contains annotated references to 'all' microprocessors, 'all' microcomputers, and 'most' microcomputing magazines. The inverted commas are there because — as the author admits candidly in his introduction — any such work is likely to be out of date even before it is published. Fat (445 pages) with minimalist presentation (the whole book is dumped directly from a word processor onto a dot-matrix printer) you'll find this a useful work if you want a ready reference to chips, computers and the ever-growing field of specialist magazines.

Computer Resource Book — Algebra
(Thomas Dwyer and Margot Critchfield, Houghton Mifflin).
Dwyer and Critchfield have clocked up an enviable string of successful computer books, and this one, part of a series, shows why. With simple, but valuable programs, the authors lead the reader (who can be a secondary student, or an instructor) through most of the phrases of the BASIC programming language which are common to all low-priced computers, and most educational timesharing systems.

Apple II BASIC
(David Goodfellow, Tab Books Inc.).
Attractively packaged, this book is clearly laid out, with an abundance of example programs; it takes a commendable approach to the business of teaching programming, with the qualities of 'programming style' introduced without fanfare. In the crowded field of 'how to program your Apple' books, this one stands out. Much of the material presented is applicable to any microcomputer.

Pre-Computer Activities
(Dorothy Diamond, Hulton Educational).
This practical guide for teachers and parents can help make children familiar with essential computer processes and language before they have hands-on ex-

perience. The book contains a number of interesting activities, including investigating binary numbers using little lights, and working with cardboard 'calculators' before getting to the real thing. The discussion on computer graphics is enlivened by reference to the solid blocks which make up a 'Pacman' figure.

Word Processing Experience
(Janet Pigott and Roger Atkins-Green, Stanley Thornes Publishers Ltd.).
Designed for schools, but ideal for adapting if you'd like to increase your skill with a word processor (or simply because you'd like to see what word processors can do so you can write one for your own microcomputer), this book looks at the mechanics of word-processing, while passing on a great deal of useful information about word-processing techniques.

An Introduction to Micro-electronics and Micro-processor Systems
(G H Curtis and P G Wilks, Stanley Thornes Publishers Ltd.).
This work was written for junior college students and older school pupils, as well as for non-specialists who wanted a comprehensive — if dry — technical introduction to the subject. The going is not easy, but it's worth the effort. Topics covered include 'Logic', 'Programming the Microcomputer' and 'Analogue, Binary and Digital Systems'.

Computer Images — State of the Art
(Joseph Deken, Thames and Hudson).
This is a beautiful book, large and glossy, and packed with quality full-colour computer-generated (or, in some cases, computer-modified) images. The whole fascinating field of modern computer graphics is discussed — from television programme introductions using photographs which are colour-modified, twisted and tweeked, to the use of incredible high-resolution images in

simulators for flight training and tank manoeuvring. You'll read (and see) how computers are used to produce images, how these are used for education and communication, why 'art for art's sake' is a goal worth pursuing, and how computer images can evolve using processes uncannily akin to the processes by which groups of cells multiply and divide. If you want to see what can be done with high resolution graphics and when time, money and skill abound, you should get this book.

Computer Bluff
(Stephen Castell, Quartermaine House Ltd.).
A much more valuable book than its title indicates, it contains a lot of information on the what and how of computers, along with a generous dollop of computer jargon (or 'How to Cheat in Computer-Speak'). The style is gentle and amusing, with no appalling puns or excessive asides (such as 'didja get that joke, buster?'). A pleasant, painless book which you can digest, then give to a parent.

Penguin Books has moved into the computer field with enthusiasm. As well as a 'Getting the Most Out of Your...' series, they have a number of games books. Two which stand out are **The Penguin Book of VIC 20 Games** (Paul Copeland) and **The Penguin Book of Commodore 64 Games** (Robert Young and Paul Copeland). Priced at £4.95 each, these large format books include such programs as 'Space Venture', 'Oil Rig' and 'Red Alert'. Worth buying, even if you do not have a VIC or a Commodore 64, simply as a source of ideas for new programs to create on your own microcomputer.

Arcade Games for Your VIC 20 and **Arcade Games for Your Commodore 64** (Brett Hale, Corgi/Addison-Wesley) by contrast, are definitely only for those who have the machine specified. The programs are locked irrevocably to the computer named. Taking advantage of a number of machine-specific features (such as sprite graphics on the 64), Brett has produced a selection of around 20 programs for each machine. Each one is listed twice, the first time for the joystick and the second time for the keyboard. Titles include 'Galaxy Robbers', 'Bullet Heads' and 'Yackman'.

CREATING ADVENTURE PROGRAMS

There are a number of books, some of which are aimed at computer owners, which will help you if you are one of the many, many computer games players who are interested in developing 'Adventure' and 'Dungeons' type programs. The place to start is with TRS Hobbies' **Dungeons and Dragons** (TM) Basic Set, which comes with the introductory rule book, Dungeon Dice (tm) and an instruction module, along with a sample scenario 'The Keep on the Borderlands'. If you're new to the field, you should start with this set to give you an idea how 'real life' Adventure programs are built up.

Additional information is provided by **Fantasy Role-Playing Games** (J. Eric Holmes, Hippocrene Books Inc.) which looks at the whole field and, despite some disparaging things to say on computer versions of such games, is worth looking for. Another overview of the field — with more sympathetic comments on the use of computers — is provided by **Dicing With Dragons — An Introduction to Role-Playing Games** (Ian Livingstone, Routledge and Kegan Paul), which includes a full 'solo Adventure', a review of the major games on the market, and a fascinating chapter on the pleasures and perils of being Dungeon Master in 'Playing God'.

Fantasy Wargaming (compiled Bruce Galloway, published Patrick Stephens) provides a complete unified system for 'historically accurate' (or at least in tune with the beliefs and circumstances of individuals in the peasant, feudal-economy times in which many Adventures are set) games. The fight, weapon and monster tables alone are worth the book, as many of their ideas can easily be incorporated into your Adventures.

There are two computer Adventure books which you could get to help you in the fascinating area of producing Adventure games on your machine.
Creating Adventure Programs on Your Computer (Andrew Nelson, Interface Publications).
Written by the author of *More Games for Your VIC20* and *Games for Your TI 99/4A*, in the Virgin Books games series, this book takes you through the task of developing an Adventure program of your own, concentrating more on the 'Loot and Pillage' school of gaming than the Scott Adams' 'solve this puzzle to advance' field. Three complete Adventure programs are included.
Write Your Own Adventure Programs for Your Microcomputer (Jenny Tyler and Les Howarth, Usborne) is a much quicker introduction to the field than Nelson's, but nevertheless packs a lot of valuable information into its 48 pages. Step-by-step instructions are provided for creating an Adventure from scratch. A complete program — 'Haunted House' — is included in the book.

The Age of Computers is the general title of four fine books produced by Wayland Publisher Limited. Each priced at £4.95, the books present a careful, but inviting, view of four aspects of the computer field, one on the history of computers and the others looking at specific areas of modern computer application. Each book is by Ian Litterick and Chris Smithers. The four titles are **The Story of Computers,** with Charles Babbage and Uncle Sir Clive Sinclair just inside the cover (and these two pictures accurately sum up the historical period covered by the book); **How Computers Work** (with chapter headings including 'Bits, Bytes and Binary', 'Decision-making by Transistor', and 'Talking With Computers'); **Computers in Everyday Life** (such things as 'Robots in the Home', 'Magnetic Money' and 'Medicine and the Disabled');

and **Computers and You** ('Computopia', 'Big Brother', 'War and Peace' and — a fascinating final chapter — 'Will Computers Need Us?').

Inside BASIC Games
(Richard Mateosian, Sybex).
This book is a slightly overwritten guide to understanding computer games. You'll learn how to write interactive programs in BASIC and how the principles of system development are applied to small computers. The book also looks at how the features of specific small computer systems have been supported in BASIC. If you can contend with the verbiage, you'll find this book well worthwhile.

1001 Things to Do With Your Personal Computer
(Mark Sawush, Tab Books).
Big and fat, and full of ideas, you'll find much here of interest to enlarge your computer horizons. The book tells you about writing music and stories with your computer, aiding a mechanic or a carpenter, solving simultaneous equations, astrology and much, much more.

Stimulating Simulations
(C.W. Engel, Hayden Book Company).
Here are 12 unique programs written in a good, general version of BASIC. The fascinating programs include 'Forest Fire', 'Rare Birds' and 'The Devil's Dungeon'. You're sure to enjoy playing those three, along with 'Diamond Thief', in which the computer decides who has committed the crime, then challenges you to discover which of the suspects is guilty. The material in this book is generally tightly programmed, and can be a helpful source of ideas to improve your own computer work.

The BASIC Handbook
(David A. Lien, Compusoft Publishing).
This is an encyclopedia of the BASIC language. It comes into its own when you find a program in a magazine or

book which you'd love to try, but are frustrated because it is written for another version of BASIC. Every BASIC word you've ever heard of (and many you may not have, such as NE, GOTO-OF and LE) is in here, along with a number of variations, one of which will almost certainly be on your machine.

BASIC Computer Games
(David Ahl, Creative Computing Press).
This is a classic work, still selling well despite the fact it was one of the first such books — if not *the* first — on the market. David Ahl has been in personal computers even before there were such things. Although several of the games are overly-dependent on the random number generator, you'll find there are many, many games you'll want to adapt and improve for your own computer.

How to Buy (and Survive) Your First Computer
(Carolee Nance Kolve, McGraw-Hill Book Company).
When is a business ready for a computer? How do you make an intelligent, informed choice among the hundreds of computers available? Will a computer improve a company's operations? Answers to these and a score of similar questions are in this book, which explains in detail what to consider before buying, how to select the right computer, and what to do after ordering the computer to ensure a successful installation. Ms Kolve has over 15 years computer experience (including a stint with IBM) and brings her experience to bear in a relatively easily-digestible guide.

Your First BASIC Program
(Rodnay Zaks, Sybex).
This book, liberally illustrated with large red dinosaurs in a variety of situations vaguely related to the text (one, for instance, as a cowboy getting tangled up in his ropes with the caption 'Be careful when looping'), is a gentle and worthwhile introduction to the not-so-secret secrets of programming in BASIC. When you want to move

beyond just typing in other people's programs from books and magazines, this may be a good place to start.

This bibliography was compiled by the series editor, Tim Hartnell, who has felt constrained not to recommend any of his own books. However, he asked us to mention two which could be of use and interest to you.

The first is **The Personal Computer Guide** (Virgin Books) which explains what a personal computer is, and answers questions like "Will it help my kids?", "What sort of games can we play on it?" and "What can I use it for in the home?". The book describes many of the most popular computers available today, with illustrations, technical specifications and other information to help you to choose the equipment best suited to your requirements. Also included is an introduction to BASIC programming, with details of programs suitable for use in the home, a list of suppliers and user clubs, and a guide to further reading. There are also chapters covering the personal computer's history and its future. When you're ready to upgrade, you'll find this book a good, unbiased, reference work which looks at the choices facing you.

Tim Hartnell's Giant Book of Computer Games.
Described by *Personal Computer News* as 'a good source of ideas', this 386-page book, published by Fontana, for £3.95, contains over 40 programs which will run with minimum modifications on most popular microcomputers. The games include chess (of a sort!), a 17K Adventure and 'Hyperwar'.

The Virgin Computer Games Series

GAMES FOR YOUR TI 99/4A

Twenty challenging programs, each one especially written for the series and guaranteed to provide hours of entertainment.

The games include KAMIKAZE PILOT (fly your tiny plane through a twisting, turning tunnel); CHALLENGER (test your reactions – dodge the alien spacecraft!); DIAMOND FEVER (fight the dreaded Vogon and mine your way through an asteroid); 3-D MAZE (stunning graphics provide a mansion for you to explore); RAM BLASTER (can you destroy all the computer components on the screen?); and DARING DAMON (this vivid program creates a hi-res horse race).

GAMES FOR YOUR TI 99/4A will improve your programming skills as you follow the instructions to put each of the programs into your machine, and comes complete with a brief dictionary of computer terms and some hints on how to extend the programs in the book. All the programs except 'Challenger' (which needs extended BASIC) will run on the standard machine.

Programs of originality and quality for all the family.

Virgin

ISBN 0 86369 030 0
United Kingdom £2.95
Australia $9.95 (recommended)